

TITLE 'COLLEEN CALCULATOR, BY C SHAW'

```

0000      ASMBL  =      0      ; 1=> ASSEMBLE THIS SECTION. 0=> THIS STUFF HAS BEEN REMOVED
;
;      ATARI CALCULATOR CARTRIDGE  COPYRIGHT 1979
;      WORK STARTED 2/20/79
;      PROGRAM STARTED 3/14/79
;
;      OPERATING SYSTEM EQUATES
;
E456      CIOV   =      $E456      CENTRAL INPUT OUTPUT ROUTINE
E45C      SETVBV =      $E45C      ; SET SYSTEM TIMERS ROUTINE
;
;      COMMAND CODES FOR IOCB
0003      OPEN  =      3      OPEN FOR INPUT/OUTPUT
0007      GETCHR =      7      GET CHARACTER(S)
0008      PUTCHR =      $B      ; PUT CHARACTER(S)
000C      CLOSE =      $C      ; CLOSE DEVICE
;
0001      SUCCES =      $01      SUCCESSFUL OPERATION
0003      EOF    =      $03      ; END OF FILE (NOT REALLY AN ERROR)
0010      IOCBSZ =      16      ; NUMBER OF BYTES PER IOCB
;
;      **=$
0008      WARMST **=$+1      ; WARM START FLAG
0009      BOOT?  **=$+1      ; SUCCESSFUL BOOT FLAG
;
0011      BRKKEY **=$+1      ; BREAK KEY FLAG
;
0052      LMARGN **=$+1      ; LEFT MARGIN (0 MIN.)
0053      RMARGN **=$+1      ; RIGHT MARGIN (39 MAX.)
0054      ROWCRS **=$+1      CURSOR COUNTERS
0055      COLCRS **=$+2
;
;      **=$22A
022A      CDTMF3 **=$+1      ; COUNT DOWN TIMER 3 FLAG
;
02F0      CRSINH **=$+1      CURSOR INHIBIT (00 = CURSOR ON)
;
;      **=$340
0340      IOCB   **=$      I/O CONTROL BLOCKS
0340      ICHID  **=$+1      HANDLER INDEX NUMBER (FF = IOCB FREE)
0341      ICDNO  **=$+1      DEVICE NUMBER (DRIVE NUMBER)
0342      ICCOM  **=$+1      COMMAND CODE
0343      ICSTA  **=$+1      STATUS OF LAST IOCB ACTION
0344      ICBAL  **=$+1      BUFFER ADDRESS LOW BYTE
0345      ICBALH **=$+1
0346      ICPTH  **=$+1      PUT BYTE ROUTINE ADDRESS - 1
0347      ICBLL  **=$+1
0348      ICBLH  **=$+1      BUFFER LENGTH LOW BYTE
0349      ICAX1  **=$+1
034A      ICAX2  **=$+1      AUXILIARY INFORMATION FIRST BYTE
034B      ICSPR  **=$+4
034C      ;
;
;      FLOATING POINT SUBROUTINES
;
0006      FPREC  =      6      ; FLOATING PT PRECISION (# OF BYTES)
;
;      IF CARRY USED THEN CARRY CLEAR => NO ERROR, CARRY SET => ERROR
D800      AFP    =      $D800      ; ASCII->FLOATING POINT (FP)
;      INBUFF+CIX -> FRO, CIX, CARRY

```

D8E6	FASC	=	\$D8E6	; FP -> ASCII	FRO-> LBUFF (INBUFF)
D9AA	IFP	=	\$D9AA	; INTEGER -> FP	
				0-\$FFFF (LSB,MSB) IN FRO,FRO+1->FRO	
D9D2	FPI	=	\$D9D2	; FP -> INTEGER	FRO -> FRO,FRO+1, CARRY
DA60	FSUB	=	\$DA60	; FRO <- FRO - FR1	,CARRY
DA66	FADD	=	\$DA66	; FRO <- FRO + FR1	,CARRY
DADB	FMUL	=	\$DADB	; FRO <- FRO * FR1	,CARRY
DB2B	FDIV	=	\$DB2B	; FRO <- FRO / FR1	,CARRY
DD89	FLDOR	=	\$DD89	; FLOATING LOAD REGO	FRO <- (X,Y)
DD8D	FLDOP	=	\$DD8D	; " " "	FRO <- (FLPTR)
DD98	FLD1R	=	\$DD98	; " " REG1	FR1 <- (X,Y)
DD9C	FLD1P	=	\$DD9C	; " " "	FR1 <- (FLPTR)
DDA7	FSTOR	=	\$DDA7	; FLOATING STORE REGO	(X,Y) <- FRO
DDAB	FSTOP	=	\$DDAB	; " " "	(FLPTR)<- FRO
DDB6	FMOVE	=	\$DDB6	; FR1 <- FRO	
DD40	PLYEVL	=	\$DD40	; FRO <- P(Z) = SUM(I=N TO 0) (A(I)*Z**I)	CARRY
				INPUT: (X,Y) = A(N),A(N-1)...A(0) -> PLYARG	
				ACC = # OF COEFFICIENTS = DEGREE+1	
				FRO = Z	
DDC0	EXP	=	\$DDC0	; FRO <- E**FRO = EXP10(FRO * LOG10(E))	CARRY
DDCC	EXP10	=	\$DDCC	; FRO <- 10**FRO	CARRY
DECD	LOG	=	\$DECD	; FRO <- LN(FRO) = LOG10(FRO)/LOG10(E)	CARRY
DED1	LOG10	=	\$DED1	; FRO <- LOG10 (FRO)	CARRY

THE FOLLOWING ARE IN BASIC CARTRIDGE:

	SIN	=	\$BD81	; FRO <- SIN(FRO) DEGFLG=0 =>RADS, 6=>DEG.	CARRY
	COS	=	\$BD73	; FRO <- COS(FRO)	CARRY
	ATAN	=	\$BE43	; FRO <- ATAN(FRO)	CARRY
	SQR	=	\$BEB1	; FRO <- SQUAREROOT(FRO)	CARRY

;FLOATING POINT ROUTINES ZERO PAGE (NEEDED ONLY IF F.P. ROUTINES ARE CALLED)

		**=\$D4	
00D4	FRO	***+FPREC	; FP REGO
00DA	FRE	***+FPREC	
00E0	FR1	***+FPREC	; FP REG1
00E6	FR2	***+FPREC	
00EC	FRX	***+1	; FP SPARE
00ED	EEXP	***+1	; VALUE OF E
00EE	NSIGN	***+1	; SIGN OF #
00EF	ESIGN	***+1	; SIGN OF EXPONENT
00F0	FCHRFLG	***+1	; 1ST CHAR FLAG
00F1	DIGRT	***+1	; # OF DIGITS RIGHT OF DECIMAL
00F2	CIX	***+1	; CURRENT INPUT INDEX
00F3	INBUFF	***+2	; POINTS TO USER'S LINE INPUT BUFFER
00F5	ZTEMP1	***+2	
00F7	ZTEMP4	***+2	
00F9	ZTEMP3	***+2	
00FB	RADFLG	***+1	; 0=RADIANS, 6=DEGREES
00FC	FLPTR	***+2	; POINTS TO USER'S FLOATING PT NUMBER
00FE	FPTR2	***+2	

;FLOATING PT ROUTINES' NON-ZERO PAGE RAM (NEEDED ONLY IF F.P. ROUTINES CALLED)

		**=\$57E	
057E	LBPR1	***+1	; LBUFF PREFIX 1
057F	LBPR2	***+1	; LBUFF PREFIX 2
0580	LBUFF	***+128	; LINE BUFFER
05E0	PLYARG	=	LBUFF+\$60 ; POLYNOMIAL ARGUMENTS
05E6	FPSCR	=	PLYARG+FPREC
05EC	FPSCR1	=	FPSCR+FPREC


```
0580      LBUF      ***+128      ; LINE BUFFER
05E0      PLYARG    =          LBUF+$60      ; POLYNOMIAL ARGUMENTS
05E6      FPSCR     =          PLYARG+FPREC
05EC      FPSCR1    =          FPSCR+FPREC
```

COLLEEN CALCULATOR, BY C SHAW

```
05E6      FSCR      =          FPSCR
05EC      FSCR1     =          FPSCR1
```

```

;
D200      AUDF1     =          $D200      COLLEEN REGISTER EQUATES
D201      AUDC1     =          AUDF1+1    ; SOUND REG 1 FREQUENCY
D20A      RANDOM    =          AUDF1+10   ; SOUND REG 1 CONTROL
; 8 BIT RANDOM NUMBER
```

UNIVERSAL EQUATES

```

;
0004      INPUT  =      4      CIO COMMANDS
0008      OUTPUT =      8      ; AUX1 ON OPEN
                                ; AUX1 ON OPEN

;
001B      ESC    =      $1B    SPECIAL CHARS IN ATARI EXTERNAL ASCII
001C      UPAROW =      ESC+1  ; ESCAPE
001D      DNAROW =      ESC+2  ; UP ARROW (CONTROL CHAR)
001E      LFAROW =      ESC+3  ; DOWN
001F      RTAROW =      ESC+4  ; LEFT
007D      CLS    =      $7D    ; RIGHT
007E      BACKSP =      CLS+1  ; CLEAR SCREEN
007F      TAB    =      CLS+2  ; BACKSP
009B      CR     =      $9B    ; CARRIAGE RETURN
009C      DELLIN =      CR+1   ; DELETE LINE
009D      INSLIN =      CR+2   ; INSERT LINE
009E      CLRTAB =      CR+3   ; CLEAR TAB
009F      SETTAB =      CR+4   ; SET TAB
00FE      DELCHR =      $FE    ; DELETE CHAR
00FF      INSCHR =      DELCHR+1 ; INSERT CHAR

```

FP PACKAGE EQUATES FOR SIN, COS, ATAN, AND SQR ROUTINES ETC

```

;
000B      NATCF  =      $B      ; NUMBER OF ATAN COEFFICIENTS FOR POLYNOMIAL EVALUATION
0006      NSCF   =      6      ; NUMBER OF SIN COEFFICIENTS

D920      XEFORM =      $D920  ; !EFORM PROCESS E FORMAT FOR FP -> ASCII CONVERSION
DA51      INTLBF =      $DA51  ; INIT LBUFF INTO INBUFF FOR FP -> ASCII CONVERSION
DC00      NORM   =      $DC00  ; NORMALIZE FLOATING POINT NUMBER - USED BY STRUNC ONLY
DE03      EXP1   =      $DE03  ; MIDDLE OF EXP10 WHERE PLYEVL IS CALLED
DE12      EXP11  =      $DE12  ; AFTER PLYEVL IN EXP10
DE89      LOG10E =      $DE89  ; LOGTEN(E) = .4342944819
DE95      XFORM  =      $DE95  ; FRO <- (FRO-(X,Y)) / (FRO+(X,Y))
DF6C      FHALF  =      $DF6C  ; FLOATING POINT CONSTANT .5
DFAE      ATCOEF =      $DFAE  ; ATAN COEFFICIENTS
DFEA      FP9S   =      $DFEA  ; FLOATING POINT CONSTANT .9999999999 (ALMOST 1)
DFF0      PIOV4  =      $DFF0  ; FLOATING POINT CONSTANT PI/4 = .7853981634

```


; CALCULATOR EQUATES

```

0001      LMARG  =      1      ; LMARGN VALUE
0026      RMARG  =      38
0026      LINLEN =      38      ; LENGTH OF LINE ON SCREEN

0016      ROWCMD =      22      ; ROWCRS FOR COMMANDS
0016      COLCMD =      22      ; COLCRS FOR COMMANDS
0001      ROWSTT =      1      ; ROW # FOR STATUS
0005      ROWREG =      5      ; ROW FOR STACK, MEM REGS
0010      ROWSCR =      16      ; TOP ROW FOR SCROLLING

0000      SIOCB  =      0*IOCBSZ ; SCREEN IOCB # (SET UP BY OS)
0010      KIOCB  =      1*IOCBSZ ; KEYBOARD IOCB #
0020      PIOCB  =      2*IOCBSZ ; PRINTER IOCB #
0030      TIOCB  =      3*IOCBSZ ; TEMP IOCB # (USED FOR FILE I/O)

000D      TOKCLN =      TOKEND-TOKCHR-1 ; LENGTH OF TOKCHR-1
0087      SLASH  =      STAR+1
0088      PLUS   =      STAR+2
0089      MINUS  =      STAR+3
008A      LPAR   =      STAR+4
008B      RPAR   =      STAR+5
008C      EQUAL  =      STAR+6
008D      LPAD   =      STAR+7
008E      NUMBER =      STAR+8

000E      PSPEC  =      14      ; PRIORITY OF SPECIAL 0-VAR FNS (PRINT, ETC.)
000D      PHIGH  =      13      ; PRIORITY OF SINGLE VAR FNS.
000A      PSPEC2 =      10      ; SPECIAL 2-VAR FNS
0009      PPOWER =      9       ; POWER, ROOT
0008      PTIMES =      8       ; * /
0007      PPLUS  =      7       ; + -
0006      PAND   =      6
0005      POR    =      5
0002      PLRPAR =      2       ; ( )
0001      PEQUAL =      1       ; =
0000      PLPAD  =      0       ; BOTTOM OF STACK SYMBOL

000E      NUMLN  =      14      ; LENGTH OF NUMBER IN ASCII FORMAT
002A      FPSLEN =      42      ; LENGTH OF FPSTK IN FP NUMBERS
0100      OPSLEN =      256     ; LENGTH OF OPSTK IN OPERANDS
0064      MEMLN  =      100     ; LENGTH OF MEMORY AREA IN FP NUMBERS
0028      TOKLEN =      LINLEN+2 ; LENGTH OF TOKBUF IN CHARS
0400      PRGLEN =      1024    ; PROGRAM MEMORY LENGTH IN BYTES
000A      SPCLEN =      SPCEND-SPCTBL-1 ; LENGTH OF SPCTBL - 1

; GRADN =      12      ; DEGREE FLAG SETTING FOR GRAD

; COLUMN NUMBERS FOR LINE 0 STATUS DISPLAY

0002      DALG   =      2
0007      DDEG   =      7
0008      DDEC   =      11
0013      DBITS  =      15+4
0019      DFIX   =      22+3
001B      DFVDUE =      27
0021      DENTER =      33

```

RAM PAGE ZERO

```

0080      ZROPG      **$B0
0080      LFRT      ***+1      ; 1 => LEFT NIBBLE, 0=> RIGHT NIBBLE (USED BY LDNIB)
0081      TOKCOD     ***+1      ; TOKEN CODE
0082      TOKPTR     ***+2      ; POINTER TO NEXT TOKBUF LOC
0083      TOKTMP     ***+2      ; LAST 0-2 CHARS READ AND SAVED
0084      TOKTIN     ***+1      ; INDEX TO TOKTMP (0-2)
0085      DHOF LG     ***+1      ; 0=> DEC, 1=>HEX, 8=>OCT, 2=>BIN
0086      KEYCHR     ***+1      ; CURRENT KEY CHAR (0-?)
0087      KEYLEN     ***+1      ; LENGTH OF CURRENT KEY WORD
0088      KEYLN2     ***+1      ; KEY LENGTH (MODIFIED BY LDCHR FOR 2 NIBBLE CHARS)
0089      LDNBSV     ***+1      ; REG Y SAVED FOR LDNIB
0090      CLRPTR     ***+1      ; FOR RAM CLEAR
0091      PKPTR      ***+2      ; POINTER TO PACKED CHAR STRING (USED BY LDNIB)
0092      KYLFRT     ***+1      ; LFRT FOR BEGINNING OF WORD
0093      KEYCNT     ***+1      ; KEY WORD NUMBER
0094      JMPTR1     ***+2      ; INDEX FOR SUBROUTINE JUMP
0095      JMPTR2     ***+2
0096      RPNALG     ***+1      ; 0=>RPN, 1=>ALG, 2=>ALGN
0097      ALGP       =      1      ; ALGEBRAIC, OPERATOR PRECEDENCE
0098      ALGNOP     =      2      ; ALGEBRAIC, SAME PRECEDENCE FOR 2 VAR OPERATORS
0099      PRNFLG     ***+1      ; 1=>PRINT
009A      OPFLG     ***+1      ; 1=>PREVIOUS TOKEN WAS AN OPERATOR
009B      NOPFLG    ***+1      ; NEW OPFLG 1=> CURRENT TOKEN IS OPERATOR
009C      PREVOP    ***+1      ; PREVIOUS OPERATOR TOKEN CODE
009D      PRVPRI    ***+1      ; " " PRIORITY (PRECEDENCE)
009E      CURPRI    ***+1      ; CURRENT " "
009F      FPSTK     ***+1      ; FPSTK POINTER (STARTS AT 0)
00A0      OPSTK     ***+1      ; OPSTK " " " "
00A1      BITINT    ***+1      ; 1-32: NUMBER OF BITS IN OCTAL & HEX ARITHMETIC
00A2      DAYTMP    ***+1      ; TEMP VAR FOR DAYSUB
00A3      CHRIND    ***+1      ; INDEX INTO CHRTAB
00A4      COUNT     ***+1      ; TEMP COUNT VAR
00A5      LDP       ***+1      ; FOR SAND, SOR, SXOR 0=>AND, 1=>OR, 2=> XOR
00A6      SHFFLG    ***+1      ; FOR SRSHF,SLSHF 0=>LEFT, 1=>RIGHT
00A7      TO        ***+1      ; TEMP VAR (ALL OF ABOVE)
00A8      T1        ***+1      ; TEMP VAR USED IN SCLINT&SCLSTA
00A9      NEGFLG    ***+1      ; PL=> POSITIVE, MI=>NEGATIVE NUMBER
00AA      INTFLG    ***+1      ; 0=>X & Y BOTH INTEGER IN ROOT, POWER
00AB      NUMFLG    ***+1      ; 1=> PREVIOUS THING DISPLAYED WAS A NUMBER
00AC      MEMNUM    ***+1      ; MEMORY NUMBER

00AD      BITBIN    ***+4      ; 2^(BITINT-1)-1
00AE      BITBN2    ***+4      ; (2^BITINT)-1
00AF      BINMIN    ***+4      ; -(2^(BITINT-1)) MSB-LSB = COMP(BITBIN)
00B0      BINARY    ***+4      ; FRO IN BINARY FORMAT
00B1      BIN2      ***+4      ; SECOND BINARY #

00B2      QUADFLG   ***+1      ; SIN QUADRANT FLAG

00B3      PC        ***+2      ; PROGRAM COUNTER LSB,MSB (INIT TO PRGMEM)
00B4      EXEC      =      2
00B5      STOPRG    =      1
00B6      PRDG      ***+1      ; 0=>IMMEDIATE MODE, 1=>STORING PROG, 2=> EXECUTING
00B7      TRACE     ***+1      ; 1 => TRACE ON (DISPLAY ALL PROGRAM EXECUTION)
00B8      DSPFLG    ***+1      ; 1=> DO NOT DISPLAY OR PRINT ANYTHING (PROGRAM EXECUTING)
00B9      SSTFLG    ***+1      ; 1=>DO SINGLE STEP (EXECUTE ONE INSTRUCTION)

```


COLLEEN CALCULATOR, BY C SHAW

```

00BF      SEFORM    ***+1      ; 1=>EFORM
00C0      FIXNUM    ***+1      ; FIX 0-9
00C1      MANTLN     ***+1      ; LENGTH OF MANTISSA
00C2      SSGN      ***+1      ; BIT 8 IS SIGN BIT
00C3      MSMD       ***+1      ; SAVE MSD OF FRO (FRO+1)
00C4      LDCAV      ***+1      ; SAVE PART CHAR FOR LDCAR
00C5      DUEFLG     ***+1      ; 0 => ANNUITY DUE/FV, 1=>ORDINARY ANNUITY/FV
                                ; 2 => ANNUITY DUE/PV, 3 => ORDINARY ANNUITY/PV
                                ; $B0 => COMPOUND INTEREST, NOT ANNUITY
00C6      ENTFLG     ***+1      ; 0 => ENTER VALUE, 1 => FIND VALUE (FOR INTEREST EQNS.)
00C7      ERRFLG     ***+1      ; 1=>ALREADY HAVE DISPLAYED ERROR MSG, DON'T DO MORE
00C8      MEMFLG     ***+1      ; 0=>ADD (SIGMA PLUS), 1=>SUB (SIGMA MINUS)
00C9      CALPTR     ***+1      ; POINTER TO CALSTK
00CA      CONFLG     ***+1      ; CONVERSION MSG LSB      0=> NO CONVERSION
00CB      SCNFLG     ***+1      ; SAVED CNFLG FROM PREVIOUS LOOP

00CC      OPSADR     ***+2      ; OPSTK ADR
00CE      MEMADR     ***+2      ; MEMORY ADR = OPSADR+$100
00D0      PRGADR     ***+2      ; PRGMEM ADR = MEMADR+$300
00D2      PC1MAX     ***+1      ; MAX_PC+1 VALUE = PRGADR/256 + 3
00D3      PC1MX1     ***+1      ; PC1MAX+1

                                ;=FRO                                ; FLOATING POINT RAM

```

*=\$480

```

0460      CALSTK      ***$80      ;SUBROUTINE CALL STACK (128/2 = 64 CALLS DEEP)
0500      TOKBUF      ***TOKLEN    ;TOKEN STRING BUFFER
0520      BLKBUF      ***20        ;ALL BLANKS
0530      CTLRE       ***20        ;ALL CTRL R'S (HORIZ. LINES)
0550      MODFAC       ***FPREC     ;INT (Y/X) AFTER MOD
0554      FTEMP        ***FPREC     ;MY OWN TEMP F.P. REG
055C      FPX          ***FPREC     ;X REG SAVED DURING STORE PROGRAM MODE
0562      ASAVE        ***1        ;REG A SAVE LOC
0563      XSAVE        ***1
0564      YSAVE        ***1
0565      PRVSTK       ***1        ;FOR DSPSTK: PREVIOUS ROWCRS VALUE AT END OF STACK
0566      SSTOLD        ***1        ;SSTFLG FROM PREVIOUS LOOP
          **LBPR1      ;FLOATING POINT
          **$600
0600      FPSTK        **FPSLEN*FPREC** ;FLOATING POINT NUMBER STACK
          ;            DISK USES $700-$2800?
          IF          ASMBL          THESE ADDRESSES START AT 700 IF NO DISK, 3000 IF DISK

```

*=\$3000

```

      OPSTK      ***+OPSLEN
               ***-1/256+1*256
      MEMORY      **MEMLN*FPREC**
               ***-1/256+1*256
      PRGMEM      ***+PRGLEN      ;USER PROGRAM MEMORY
               .ENDIF

```


COLLEEN CALCULATOR BY C SHAW

```

**$9800
9800                                SEXPE
9800 A2 89                        LDX    #LOG10E      ;E^X (SEE SHEP ATARI BASIC $DDCO EXP)
9802 A0 DE                        LDY    #LOG10E/256    ;E^X = 10^(X*LOGTEN(E))
9804 20 C5 AD                      JSR    LD1MUL        ;FRO <- FRO*LOG10E

9807                                SEXPTE
9807 A9 00                        LDA    #0            ;10^X (SEE SHEP ATARI BASIC $DDCC EXP10)
9809 B5 F1                        STA    DIGRT         ;CLEAR TRANSFORM FLAG
980B A5 D4                        LDA    FRO          ;XFMFLG
980D B5 F0                        STA    FCHRFLG        ;SAME AS SGNFLG  REMEMBER ARG SIGN
980F 29 7F                        AND    #$7F          ;& MAKE PLUS
9811 B5 D4                        STA    FRO
9813 38                          SEC
9814 E9 40                        SBC    #$40
9816 30 18                        BMI    SEXPO5         ;X<1 SO USE SERIES DIRECTLY (BUT CHECK FOR 0 FIRST)
9818 C9 04                        CMP    #FPREC-2
981A 10 2B                        BPL    SFERR2         ;ARG TOO BIG
981C 20 BB 9F                      JSR    FPUSH0        ;SAVE ARG ON CALCULATOR FP STACK
981F 20 B3 A6                      JSR    SINTEG        ;GREATEST INTEGER <= X
9822 20 D2 D9                      JSR    FPI           ;MAKE INTEGER
9825 A5 D5                        LDA    FRO+1        ;CHECK MSB
9827 D0 1E                        BNE    SFERR2         ;SHOULDN'T HAVE ANY
9829 A5 D4                        LDA    FRO
982B B5 F1                        STA    DIGRT         ;XFMFLG          SAVE MULTIPLIER EXP
982D 20 AA D9                      JSR    IFP          ;NOW TURN IT BACK TO FP
9830 20 80 A9                      JSR    SPSUB        ;USE CALC ROUTINE  ARG FROM STACK - INTEGER PART = FRACTION PART
9833                                SEXPO5
9833 A5 D4                        LDA    FRO
9835 D0 0B                        BNE    SEXP10
9837 A9 01                        LDA    #1            ;10^0 = 1
9839 20 B9 A1                      JSR    PSET0
983C 20 12 DE                      JSR    EXP11        ;$DE12 DO 10^X, SKIPPING PLYEVL  LDA XFMFLG . . .
983F B0 06                        BCS    SFERR2
9841                                SEXPRT
9841 60                          RTS

9842                                SEXP10
9842 20 03 DE                      JSR    EXP1          ;DO REST OF 10^X
9845 90 FA                        BCC    SEXPRT        ;CC => OK => RETURN
9847                                SFERR2
9847 4C 95 A3                      JMP    BITERR       ;DISPLAY ERROR MESSAGE

```

*=\$A000

INITIALIZATION

CARTRIDGE COLD/WARM START LOC

984A	START			
984A A9 00	LDA	#0		
984C AA	TAX			; CLEAR ZERO PAGE RAM (\$80-\$FF)
984D	INIT2			
984D 95 80	STA	ZROPG.X		
984F E8	INX			
9850 10 FB	BPL	INIT2		
9852 A9 07	LDA	#07		; SET UP INDIRECT POINTERS TO RAM
9854 A6 09	LDX	BOOT?		; SUCCESSFUL BOOT?
9856 F0 02	BEG	NOBOOT		; NO.
9858 A9 30	LDA	#30		; YES. ALLOW ROOM FOR DOS IN RAM
985A	NOBOOT			
985A 85 CD	STA	OPBADR+1		; OPSTK ADR MSB
985C 18	CLC			
985D 69 01	ADC	#1		
985F 85 CF	STA	MEMADR+1		; MEMORY ADR MSB
9861 69 03	ADC	#3		
9863 85 D1	STA	PRGADR+1		; PRGMEM ADR MSB
9865 69 03	ADC	#3		
9867 85 D2	STA	PCIMAX		; END OF PRGMEM ADR MSB
9869 69 01	ADC	#1		
986B 85 D3	STA	PCIMX1		; PCIMAX+1
986D A0 01	LDY	#LMARG		; 1 SET UP MARGINS
986F 84 52	STY	LMARGN		
9871 8C F0 02	STY	CRGINH		; <0 => INHIBIT CURSOR
9874 84 94	STY	RPHALG		
9876 A9 26	LDA	#RMARG		; DEFAULT IS ALGEBRAIC WITH OPERATOR PRECEDENCE ALOP=1
9878 85 53	STA	RMARGN		
987A A2 10	LDX	#KIOCB		OPEN KEYBOARD, (SCREEN OPENED BY OS)
987C 88	DEY			; 0
987D 20 F6 AC	JSR	CIOINT		
CHECK FOR ERROR????				
9880 A9 08	LDA	#8		
9882 85 C0	STA	FIXNUM		; INIT TO FIX 8
9884 A9 8D	LDA	#LPAD		
9886 20 D7 A1	JSR	PUSHOP		; INIT OPERATOR STACK WITH LPAD ON BOTTOM
9889 A9 05	LDA	#TOKBUF/100		
988B 85 B3	STA	TOKPTR+1		
988D A5 08	LDA	WARMST		
988F D0 06	BNE	WARM		; DON'T CLEAR MEMS IF WARM START
9891 20 03 AA	JSR	MEMCLR		; CLEAR MEMORY REGISTERS
9894 20 F5 A9	JSR	SCLPD		; INITIALIZE PC TO START OF PRGMEM AND CLEAR PRGMEM TO ALL STP'S
9897	WARM			
9897 A4 D1	LDY	PRGADR+1		; INIT PC IN ANY CASE

COLLEEN CALCULATOR, BY C SHAW

```
9899 84 BA          STY      PC+1
989B A2 13          LDX      #20-1          ; INIT BLKBUF & CTLRS
989D                INIT4
989D A9 20          LDA      #'
989F 9D 28 05       STA      BLKBUF, X
98A2 A9 12          LDA      #'R-64
98A4 9D 3C 05       STA      CTLRS, X
98A7 CA             DEX
98A8 10 F3          BPL      INIT4

98AA 84 94          STY      RPNALG          ; DEFAULT IS ALGEBRAIC WITH OPERATOR PRECEDENCE  ALGP=1
                                           ; INIT SCREEN DISPLAY
                                           ; LINE 0-1  "ALG RAD . . .

98AC A9 B6          LDA      #STATLN
98AE 20 04 9C       JSR      STMSG2
98B1 20 74 A2       JSR      PUTCHS
98B4 A9 D4          LDA      #STLN2
98B6 20 04 9C       JSR      STMSG2
98B9 20 27 9C       JSR      INVID
98BC 20 74 A2       JSR      PUTCHS          ; CHANGE TOKBUF TO INVERSE VIDEO & RELOAD A, X, Y

98BF 20 A6 AA       JSR      DSPALL          ; STANDARD DISPLAY

98C2 A9 10          LDA      #16
98C4 20 F4 A4       JSR      SBITS2
```

```

98C7          LOOP      LDA      CRSINH      ; BREAK KEY HIT, CAUSING CURSOR TO BE TURNED ON?
98C7 AD F0 02      BNE      MAIN02      ; NO.
98CA D0 06
98CC EE F0 02      INC      CRSINH      ; INHIBIT CURSOR
98CF 20 93 AA      JSR      SEND        ; DISPLAY STACK, CHANGE PROG
98D2          MAIN02    LDA      PRDG
98D2 A5 B8      CMP      #STOPRG      ; STORE PROGRAM?
98D4 C9 01      BNE      NOSTOR
98D6 D0 65

```

STORE PROGRAM MODE

```

98D8 20 5A 9C      JSR      DSPRG      ; DISPLAY OLD VALUE IN PROGRAM LOC
98D8 20 5A 9C
98D8 20 51 9A      JSR      LEX        ; GET NEXT TOKEN FROM PROGRAM MEM.
98DE 20 27 A2      JSR      PUTDEL
98E1 A5 B1      LDA      TOKCOD      ; CHECK FOR SPECIAL COMMAND
98E3 A2 0A      LDX      #SPCLEN
98E5          LOOP3    CMP      SPCTBL, X
98E5 DD E4 BA      BNE      LOOP4
98E8 D0 0F
98EA A9 01      LDA      #1          ; "NUMBER" => SPECIAL COMMAND FOUND
98EC 85 A2      STA      NUMFLG      ; ALWAYS ON SEPARATE LINE.
98EE 20 31 99      JSR      DSPRG
98F1 A5 B1      LDA      TOKCOD
98F3 20 05 A0      JSR      SUBCAL      ; CALL SUBROUTINE
98F6 4C C7 98      JMP      LOOP      ; CONTINUE

98F9          LOOP4    DEX
98F9 CA          BPL      LOOP3      ; TRY NEXT ONE
98FA 10 E9
98FC C9 BE      CMP      #NUMBER      ; NOT SPECIAL COMMAND => SAVE      NUMBER?
98FE D0 24      BNE      STPR40      ; NO.
9900 20 B5 A1      JSR      PCNCHK      ; CHECK PC TO SEE IF ROOM FOR NUMBER
9903 B0 C2      BCS      LOOP        ; ERROR      END OF MEM

9905 20 A7 DD      JSR      FSTOR      ; STORE FRO IN PRGMEM
9908 A0 07      LDY      #FPREC+1
990A A9 BE      LDA      #NUMBER
990C 91 B9      STA      (PC), Y
990E A0 00      LDY      #0
9910 91 B9      STA      (PC), Y
9912 20 9D A1      JSR      PCADDN      ; MOVE PC PAST NUMBER
9915 A9 16      LDA      #22
9917 B5 54      STA      ROWCRS
9919 85 55      STA      COLCRS
991B 20 B8 9C      JSR      DG40      ; PTTXTP NUMBER
991E 20 9D A2      JSR      PUTCRP
9921 4C C7 98      JMP      LOOP

9924          STPR40    LDY      #0          ; NOT A NUMBER
9924 A0 00      STA      (PC), Y
9926 91 B9      JSR      PCINC
9928 20 99 A1

```

; DISPLAY NEW TOKEN AFTER OLD

971B 20 9B A2	JSR	PUTCRP
971E 20 9D A2	JMP	LOOP
9721 4C C7 9B		
9724	STPR40	
9724 A0 00	LDY	#0
9726 91 B9	STA	(PC), Y
972B 20 99 A1	JSR	PCINC

; NOT A NUMBER

; DISPLAY NEW TOKEN AFTER OLD

COLLEEN CALCULATOR, BY C SHAW

972B 20 31 99	JSR	DSPROG
972E	JMPLQP	
972E 4C C7 9B	JMP	LOOP

9731	DSPROG		; SET UP CURSOR AND DISPLAY COMMAND
9731 A9 16	LDA	#22	
9733 85 54	STA	ROWCRS	
9735 85 55	STA	COLCRS	
9737 20 C0 9C	JSR	PUTCMD	
973A 4C 9D A2	JMP	PUTCRP	

993D	NOSTOR			; NOT STORE PROGRAM MODE
993D 20 92 A7		JSR	DSPSTK	; DISPLAY STACK
9940 20 8B 9F		JSR	FPUSH0	; STORE OLD # IN CASE RPN
9943 20 51 9A		JSR	LEX	
9946 90 06		BCC	NOST10	
9948 20 9D 9F		JSR	FPOPO	; EXEC ERROR (OUT OF EXEC MODE) RELOAD X
994B 4C C7 98		JMP	LOOP	
994E	NOST10			
994E A5 CA		LDA	CONFLG	; SAVE CONVERSION FLAG
9950 B5 CB		STA	SCONFG	
9952 A9 00		LDA	#0	
9954 B5 97		STA	NOPFLG	
9956 A6 BE		LDX	SSTFLG	
9958 BE 66 05		STX	SSTOLD	
995B B5 BE		STA	SSTFLG	
995D A5 B1		LDA	TOKCOD	
995F C9 BE		CMP	#NUMBER	; NUMBER?
9961 D0 0A		BNE	MAIN05	; NO. SKIP
9963 A5 94		LDA	RPNALG	; RPN?
9965 F0 03		BEG	MAIN04	; YES.
9967 20 86 9F		JSR	FPOP1	; NO. DISCARD NUMBER PUSHED ON STACK
996A	MAIN04			
996A 4C 21 9A		JMP	ENDWLP	
996D	MAIN05			; NOT NUMBER
996D 20 86 9F		JSR	FPOP1	; DISCARD # STORED ON STACK IN CASE RPN
9970 A0 00		LDY	#0	
9972 B4 A2		STY	NUMFLG	
9974 A5 B1		LDA	TOKCOD	
9976 20 06 A1		JSR	GETPRI	
9979 B5 9A		STA	CURPRI	
997B C9 0D		CMP	#PHIGH	
997D 90 11		BCC	MAIN40	
997F A5 B1		LDA	TOKCOD	; SPECIAL OR HIGH
9981 20 05 A0		JSR	SUBCAL	; EXECUTE SUBROUTINE
9984 A5 9A		LDA	CURPRI	
9986 C9 0D		CMP	#PHIGH	
9988 D0 03		BNE	MAIN35	
998A 20 EE A1		JSR	FDSCOM	; HIGH
998D	MAIN35			
998D 4C 21 9A		JMP	ENDWLP	
9990	MAIN40			; A=CURRENT PRIORITY
9990 A6 94		LDX	RPNALG	
9992 D0 15		BNE	MAIN60	
9994 C9 03		CMP	#PLRPAR+1	; RPN
9996 B0 06		BCS	MAIN50	
9998 20 B5 9B		JSR	KEYERR	; () = NOT LEGAL IN RPN
999B 4C 25 9A		JMP	ENDLP3	


```

9996 B0 06      BCS      MAIN50
9998 20 B5 93      JSR      KEYERR
999B 4C 25 9A      JMP      ENDLP3

```

```
;( ) = NOT LEGAL IN RPN
```

COLLEEN CALCULATOR, BY C SHAW

```

999E          MAIN50
999E A5 B1      LDA
99A0 20 05 A0      JSR
99A3 20 EE A1      JSR
99A6 4C 25 9A      JMP

```

```

TOKCOD
SUBCAL
FDSCOM
ENDLP3

```

```
; EXECUTE SUBROUTINE
```

```

99A9          MAIN60
99A9 A6 B1      LDX
99AB E0 8A      CPX
99AD D0 08      BNE

```

```

TOKCOD
#LPAR
MAIN62

```

```
; NOT RPN
CHECK FOR 2 OPS IN A ROW
```

```
; OP CAN BE FOLLOWED BY '('
; NOT '('
```

```

99AF BA      TXA
99B0 20 D7 A1      JSR
99B3 E6 97      INC
99B5 10 6A      BPL
99B7

```

MAIN62

```

TXA
PUSHOP
NOPFLG
ENDWLP

```

```
; IS '('
; PUSH '('
; <-1 LPAR CAN'T BE FOLLOWED BY BINARY OP (EXCEPT LPAR)
; JMP
```

```

99B7 C9 03      CMP
99B9 90 02      BCC
99BB E6 97      INC
99BD
99BD A6 96      LDX
99BF F0 08      BEQ
99C1 A9 42      LDA
99C3 20 B7 9B      JSR
99C6 20 C4 A1      JSR
99C9 20 86 9F      JSR

```

MAIN65

```

#PLRPAR+1
MAIN65
NOPFLG
OPFLG
WLOOP
#TOPMSG
ERRSUB
POPOP
FPOP1

```

```
; ')' AND '=' CAN BE FOLLOWED BY OP
; 2-VAR OPERATOR
```

```
; 2 BINARY OPS IN A ROW IS ILLEGAL: IGNORE 1ST OP
```

```
; DISCARD PREV OP
; DISCARD EXTRA #
```

```

99CC          WLOOP
99CC 20 C4 A1      JSR
99CF B5 98      STA
99D1 20 06 A1      JSR
99D4 B5 99      STA

```

WLOOP

```

POPOP
PREVOP
GETPRI
PRVPRI

```

```

99D6 A5 B1      LDA
99D8 C9 8B      CMP
99DA D0 19      BNE
99DC A5 98      LDA
99DE C9 8A      CMP
99E0 F0 3F      BEQ
99E2 C9 8D      CMP
99E4 D0 06      BNE
99E6 20 D7 A1      JSR
99E9 4C 21 9A      JMP

```

```

TOKCOD
#RPAR
WLP10
PREVOP
#LPAR
ENDWLP
#LPAD
WLP05
PUSHOP
ENDWLP

```

```
; TOKCOD = RPAR '('
```

```
; PREVOP = LPAR '(' (NUMBER) => IGNORE PARENS
```

```
; LPAD ')' => PUSH LPAD BACK ON BOTTOM OF STACK
```

```

99EC          WLP05
99EC 20 05 A0      JSR
99EF 20 EE A1      JSR
99F2 4C CC 99      JMP

```

WLP05

```

SUBCAL
FDSCOM
WLOOP

```

```
; EXECUTE SUBROUTINE OP ')' => PERFORM OP & CONTINUE
```

```

99F5          WLP10
99F5 C9 8C      CMP
99F7 D0 06      BNE
99F9 A5 98      LDA
99FB C9 8A      CMP
99FD F0 CD      BEQ
99FF
99FF A5 99      LDA

```

WLP10

```

#EQUAL
WLP20
PREVOP
#LPAR
WLOOP
WLP20
PRVPRI

```

```
; '(' '=' => CONTINUE (CLOSE ALL OPEN LPARS)
; NOT (TOKCOD = EQUAL AND PREVOP = LPAR)
```

COLLEEN CALCULATOR, BY C SHAW

```

9A01 C5 9A      CMP      CURPRI
9A03 90 08      BCC      WLP30
9A05 A5 98      LDA      PREVOP      ; PRVPRI>CURPRI
9A07 20 05 A0   JSR      SUBCAL      ; EXECUTE SUBROUTINE
9A0A 20 EE A1   JSR      FDSCOM
9A0D 4C CC 99   JMP      WLOOP

9A10            WLP30            ; PRVPRI<CURPRI
9A10 A5 98      LDA      PREVOP
9A12 20 D7 A1   JSR      PUSHOP
9A15 A5 B1      LDA      TOKCOD
9A17 C9 8C      CMP      #EQUAL
9A19 F0 06      BEQ      ENDWLP      ; LPAD '=' => DONE
9A1B 20 D7 A1   JSR      PUSHOP      ; NOT '=' => SAVE CURRENT OP & PUSH STACK
9A1E 20 BB 9F   JSR      FPUSHO

9A21            ENDWLP
9A21 A5 97      LDA      NOPFLG
9A23 B5 96      STA      OPFLG
9A25            ENDLP3
9A25 AD 66 05   LDA      SSTOLD      ; SINGLE STEP?
9A28 F0 06      BEQ      ENDLP4      ; NO.
9A2A A9 00      LDA      #0          ; YES. GO BACK TO IMMEDIATE MODE
9A2C B5 BB      STA      PROG
9A2E B5 BE      STA      SSTFLG
9A30            ENDLP4
9A30 A5 CA      LDA      CONFLG      ; IS CONFLG UNCHANGED?
9A32 38         SEC
9A33 E5 CB      SBC      SCONFG
9A35 D0 02      BNE      ENDSKP2     ; CONFLG CHANGED => DO NOTHING
9A37 B5 CA      STA      CONFLG      ; CONFLG NOT CHANGED => CLEAR
9A39            ENDSKP2
9A39 4C C7 98   JMP      LOOP

;
                                END OF MAIN PROGRAM LOOP

```


LEXICAL ANALYZER

```

FETCH NEXT TOKEN FROM TERMINAL AND
SET UP TOKEN CODE IN TOKCOD, PUT STRING IN TOKBUF

```

```

9A3C          LXINIT          ;SUBROUTINE TO DISPLAY '>', SET UP CURSOR
9A3C A9 00          LDA      #0
9A3E 85 82          STA      TOKPTR
9A40 8D F0 02       STA      CRSINH          ;CURSOR ON
9A43 A9 02          LDA      #LMARG+1
9A45 85 55          STA      COLCRS
9A47 A9 17          LDA      #23
9A49 85 54          STA      ROWCRS          ;START AT BOTTOM OF SCREEN
9A4B A9 3E          LDA      #'>
9A4D 20 31 A2       JSR      PTCHR          ;TEST CHAR

```

```

9A50                                EXEC20
9A50 60                                RTS

```

```

9A51          LEX          LDA      #0          ; CLEAR FLAG => NO ERROR THIS TIME
9A51 A9 00          STA      ERRFLG
9A53 85 C7

```

```

9A55 A5 BB      LDA      PROG
9A57 C9 02      CMP      #EXEC      ;EXECUTING PROGRAM?
9A59 D0 1A      BNE      NOEXEC     ;NO
9A5B 20 27 A2   JSR      PUTDEL     ;CLEAR BOTTOM LINE - SET UP CURSOR FOR DISPLAY
9A5E 20 5A 9C   JSR      DSPRG      ;DISPLAY PROGRAM ADDR & CONTENTS IF TRACE
9A61 20 7D A2   JSR      PUTCRP     ;CR ON PRINTER ONLY
9A64 20 61 A1   JSR      NCHKLD     ;YES.  LOAD TOKEN AND CHECK FOR NUMBER
9A67 B0 E7      BCS      EXEC20     ;ERROR
9A69 D0 05      BNE      EXEC10     ;NOT NUMBER
9A6B 20 7D A1   JSR      PCADDN     ;NUMBER          MOVE PC PAST #
9A6E 18         CLC
9A6F 60         RTS
9A70                                ;IF END OF PROG MEM EXECUTE INSTRUCTION BEFORE STOPPING
                                EXEC10
9A70 20 99 A1   JSR      PCINC      ;NOT NUMBER
9A73 18         CLC
9A74 60         RTS
                                ;CLEAR ANY ERROR

```

9A75	NOEXEC			
9A75	20	3C 9A	JSR	LXINIT ; DISPLAY '>' ; INIT CURSOR
9A78	20	26 A0	JSR	GTCHR ; A=NEXT CHAR
9A7B	C9	20	CMP	#
9A7D	F0	D2	BEQ	LEX
9A7F	C9	9C	CMP	#DELLIN
9A81	F0	CE	BEQ	LEX

CHECK FOR SINGLE CHAR TOKENS

```

9A83 A2 0D          LDX      #TOKCLN
9A85                LXL P20
9A85 DD CC BA      CMP      TOKCHR, X
9A88 D0 08          BNE      LEX30

```

7ABA BD DA BA
7ABD 85 81
7ABF 4C 85 9B

LDA
STA
JMP

TOKTBL, X
TOKCOD
LXRTN2

7A92 LEX30
7A92 CA
7A93 10 F0

DEX
BPL

LXLP20

CHECK FOR KEYWORD (ALPHA)

```

9A95 C9 41      CMP    #'A
9A97 90 4C      BCC    LXNMCK
9A99 C9 5B      CMP    #'Z+1
9A9B B0 4B      BCS    LXNMCK

9A9D 20 E2 A2    JSR    UNPINT
9AA0              KEYLP1
9AA0 20 F3 A2    JSR    UNPNUM

9AA3 F0 4F      BEQ    LXERR2      ; END OF LIST IF 0 COUNT=>ERROR
9AA5              KEYLP2
9AA5 20 24 A1    JSR    LDCHR

9AAB A6 8B      LDX    KEYCHR
9AAA DD 00 05    CMP    TOKBUF,X
9AAD 90 26      BCC    KEY20      ; NO MATCH: HAVEN'T GONE FAR ENOUGH
9AAF D0 43      BNE    LXERR2      ; NO MATCH: HAVE GONE TOO FAR - GIVE ERROR MSG
9AB1 E8          INX
9AB2 B6 8B      STX    KEYCHR

9AB4 E4 B2      CPX    TOKPTR
9AB6 90 ED      BCC    KEYLP2
9AB8 F0 EB      BEQ    KEYLP2

9ABA B4 8B      STY    LDNBSV      ; NEED TO FETCH MORE CHARS FROM TERMINAL
9ABC E6 B2      INC    TOKPTR      ; SAVE PREVIOUS CHAR
9ABE 20 26 A0    JSR    QTCHR

9AC1 C9 9C      CMP    #DELLIN
9AC3 F0 8C      BEQ    LEX

9AC5 A4 8B      LDY    LDNBSV
9AC7 C9 41      CMP    #'A
9AC9 90 10      BCC    ENDL15      ; END OF CHAR STRING (INCOMPLETE KEYWORD MATCH)
9ACB C9 5B      CMP    #'Z+1
9ACD B0 0C      BCS    ENDL15
9ACF A6 8B      LDX    KEYCHR
9AD1 E4 8A      CPX    KEYLN2
9AD3 90 D0      BCC    KEYLP2      ; NOT END OF WORD => CONTINUE

9AD5              KEY20
9AD5 20 FF A2    JSR    UNPNXT      ; TRY NEXT WORD IN LIST
9ADB 4C A0 9A    JMP    KEYLP1      ; CONTINUE

9ADB              ENDL15
9ADB A5 8F      LDA    KEYCNT
9ADD 85 B1      STA    TOKCDD
9ADF 20 D7 A2    JSR    SAVCHR
9AE2 4C 7F 9B    JMP    LEXRTN

```

CHECK FOR NUMBER

9AE5	LXNMCK			
9AE5 C9 2E		CMP	#'	
9AE7 D0 0F		BNE	LXNDOT	
9AE9 A6 87		LDX	DHOF LG	; DEC?
9AEB D0 07		BNE	LXERR2	; NO. ' ' NOT ALLOWED
9AED E6 82		INC	TOKPTR	; SAVE " "
9AEF 20 A5 A0		JSR	GETDHO	; GET DEC, HEX OR OCT DIGIT (ACCORDING TO DHOF LG)
9AF2 90 1E		BCC	LXHVDT	
9AF4	LXERR2			
9AF4 A9 4C		LDA	#KEYMSG	; ' ' WITH NO DIGITS "NOT VALID COMMAND"
9AF6 D0 74		BNE	LEXERR	; JMP
9AFB	LXNDOT			
9AFB 20 B1 A0		JSR	DHOCHK	
9AFB B0 F7		BCS	LXERR2	; NO MATCH AT ALL SO ILLEGAL CHAR
				HAVE NUMBER
9AFD	LXLP40			
9AFD A6 82		LDX	TOKPTR	
9AFF E0 0F		CPX	#NUMLEN+1	; LIMIT TO NUMLEN CHARS
9B01 B0 67		BCS	LENERR	
9B03 20 A5 A0		JSR	GETDHO	
9B06 90 F5		BCC	LXLP40	; KEEP GETTING DIGITS
9B08 C9 2E		CMP	#'	
9B0A D0 11		BNE	LXND2	
9B0C A6 87		LDX	DHOF LG	; DEC?
9B0E D0 E4		BNE	LXERR2	; NO. ' ' NOT ALLOWED
9B10 E6 82		INC	TOKPTR	; SAVE " "
9B12	LXHVDT			
9B12 A6 82		LDX	TOKPTR	; CHECK FOR BUFFER OVERFLOW
9B14 E0 0F		CPX	#NUMLEN+1	; LIMIT TO NUMLEN CHARS
9B16 B0 52		BCS	LENERR	; TOO MANY DIGITS
9B18 20 A5 A0		JSR	GETDHO	
9B1B 90 F5		BCC	LXHVDT	
9B1D	LXND2			
9B1D A6 87		LDX	DHOF LG	
9B1F D0 24		BNE	LXNUM	; OCTAL OR HEX => NO EXP ALLOWED
9B21 C9 45		CMP	#'E	; DECIMAL CHECK FOR EXPONENT
9B23 D0 20		BNE	LXNUM	; NO E=> END OF NUMBER
9B25 E6 82		INC	TOKPTR	; SAVE E
9B27 20 A5 A0		JSR	GETDHO	
9B2A 90 0F		BCC	LXGT2	; HAVE DIGIT
9B2C C9 2B		CMP	#'+	; NOT A DIGIT. '+' OR '-'?
9B2E F0 04		BEQ	LX50	
9B30 C9 2D		CMP	#'-	
9B32 D0 2F		BNE	LX60	; E IS NOT FOR EXPONENT=> DONE WITH NUMBER
9B34	LX50			
9B34 E6 82		INC	TOKPTR	; E IS FOR EXPONENT => SAVE '+' OR '-'
9B36 20 A5 A0		JSR	GETDHO	
9B39 B0 B9		BCS	LXERR2	

9B34 E 02	LX50	INC	TOKPTR
9B36 20 A5 A0		JSR	GETDHO
9B39 B0 B9		BCS	LXERR2

;E IS FOR EXPONENT => SAVE '+' OR '-'

COLLEEN CALCULATOR, BY C SHAW

9B3B		LXGT2		
9B3B 20 A5 A0		JSR	GETDHO	; GET 2ND DIGIT OF EXPONENT
9B3E B0 05		BCS	LXNUM	; NO 2ND DIGIT
9B40 20 A5 A0		JSR	GETDHO	; HAVE 2ND DIGIT. IS THERE 3RD?
9B43 90 25		BCC	LENERR	; ERROR - EXPONENT TOO LARGE
9B45		LXNUM		
9B45 20 D7 A2		JSR	SAVCHR	; SAVE LAST CHAR FOR NEXT TOKEN
9B48		LXN2		
9B48 A0 00		LDY	#0	
9B4A A9 9B		LDA	#CR	
9B4C 91 82		STA	(TOKPTR), Y	
9B4E 20 2B A4		JSR	SNUMB	; ASCII -> FP
9B51 EE F0 02		INC	CRSINH	; CURSOR OFF
9B54 A5 BB		LDA	PROG	
9B56 C9 01		CMP	#STOPRG	
9B58 F0 03		BEG	LXN3	; STORE PROGRAM => NO DISPLAY
9B5A 20 57 9D		JSR	FDSP0	; DISPLAY # OTHERWISE FOR ALL LEX CALLS
9B5D		LXN3		
9B5D A9 BE		LDA	#NUMBER	
9B5F B5 B1		STA	TOKCOD	
9B61 D0 4D		BNE	LXRTN3	; JMP
9B63		LX60		
9B63 20 D7 A2		JSR	SAVCHR	; SAVE E FOR NEXT TOKEN
9B66 C6 B2		DEC	TOKPTR	
9B68 10 DB		BPL	LXNUM	; JMP
9B6A		LENERR		
9B6A A9 9D		LDA	#DIGMSG	; TOO MANY DIGITS
9B6C		LEXERR		
9B6C A0 01		LDY	#1	
9B6E 8C F0 02		STY	CRSINH	; CURSOR OFF
9B71 4B		PHA		
9B72 20 0B A2		JSR	PICRPD	; EXTRA CR SO CHARS NOT LOST
9B75 6B		PLA		
9B76 20 B7 9B		JSR	ERRSUB	
9B79 20 0B A2		JSR	PUTCR	; EXTRA CR
9B7C 4C 51 9A		JMP	LEX	; TRY AGAIN
9B7F		LEXRTN		
9B7F A0 00		LDY	#0	
9B81 A9 9B		LDA	#CR	
9B83 91 82		STA	(TOKPTR), Y	
9B85		LXRTN2		
9B85 EE F0 02		INC	CRSINH	; CURSOR OFF
9B88 A5 BB		LDA	PROG	
9B8A C9 01		CMP	#STOPRG	
9B8C F0 22		BEG	LXRTN3	; DON'T DISPLAY IF STORE PROGRAM MODE
9B8E A5 BD		LDA	DSPFLG	
9B90 D0 1E		BNE	MAIN21	; NO DISPLAY

COLLEEN CALCULATOR, BY C SHAW

```

9B92 20 47 A3      JSR      UNPKEY      ; UNPACK KEYWORD TOKNUM INTO TOKBUF
9B95 A6 A2          LDX      NUMFLG
9B97 D0 04          BNE      MAIN15
9B99 A9 13          LDA      #19        ; # OF BLANKS NEEDED TO GET COMMAND IN PROPER COL ON PRINTER
9B9B D0 06          BNE      MAIN20    ; JMP      PREVIOUS TOKEN WAS NOT A NUMBER
9B9D                MAIN15            ; PREVIOUS TOKEN WAS A NUMBER
9B9D A7 16          LDA      #ROWCMD
9B9F 85 54          STA      ROWCRS
9BA1 A9 01          LDA      #1
9BA3                MAIN20
9BA3 20 6F 9F      JSR      PUTBLK      ; PUT A BLANKS ON PRINTER ONLY

9BA6 A9 16          LDA      #COLCMD
9BA8 85 55          STA      COLCRS
9BAA 20 20 9C      JSR      TOKINT      ; SET UP X=TOKPTR, A=TOKBUF, Y=TOKBUF/256
9BAD 20 05 A2      JSR      PTTXTP
9BB0                MAIN21

```

```

9BB0                LXRTN3
9BB0 20 27 A2      JSR      PUTDEL      ; DELETE BOTTOM LINE
9BB3 18            CLC                ; NO ERROR
9BB4                INIT              ; POWER UP INIT: JUST RETURN
9BB4 60            RTS

```

END OF LEX

COLLEEN CALCULATOR, BY C SHAH

98B5	KEYERR			
98B5 A9 4C		LDA	#KEYMSG	
98B7	ERRSUB			
				OUTPUT "ERROR - " ; MESSAGE A=LSB OF ADDRESS
				RETURN CS => ERROR
98B7 A6 C7		LDX	ERRFLG	
98B9 D0 1A		BNE	ERRRTN	; RETURN IF ERROR ALREADY DISPLAYED
98BB E6 C7		INC	ERRFLG	; <- 1 SET FLAG
98BD AA		TAX		
98BE A5 54		LDA	ROWCRS	
98C0 48		PHA		; SAVE OLD CURSOR LOC SO IT CAN BE RESTORED LATER
98C1 A5 55		LDA	COLCRS	
98C3 48		PHA		
98C4 8A		TXA		
98C5 48		PHA		
98C6 20 D7 98		JSR	ERRSB2	; DO SOUND, SET UP TO OUTPUT "ERROR - "
98C9 68		PLA		; RELOAD MESSAGE ADDR
98CA A0 8C		LDY	#ERRTBL/256	
98CC 20 F2 98		JSR	PTMSG2	; PUT TEXT ON SCREEN AND PRINTER
98CF 68		PLA		
98D0 85 55		STA	COLCRS	
98D2 68		PLA		
98D3 85 54		STA	ROWCRS	
98D5	ERRRTN			
98D5 38		SEC		
98D6 60		RTS		
98D7	ERRSB2			
98D7 A9 00		LDA	#0	
98D9 85 BD		STA	DSPFLG	; TURN DISPLAY ON
98DB A5 BB		LDA	PROG	
98DD 29 01		AND	#1	
98DF 85 BB		STA	PROG	; STOP EXECUTION, IF ANY
98E1 A2 01		LDX	#1	
98E3 8E F0 02		STX	CRSINH	; CURSOR OFF
98E6 A9 80		LDA	##80	; OUTPUT ERROR SOUND
98E8 20 3A 9C		JSR	SOUND	
98EB 20 97 A2		JSR	PTCRPN	; PUT CR ON PRINTER IF PREVIOUS WAS NUMBER
98EE A9 80		LDA	#ERRMSG	PUT "ERROR"
				; DISPLAY PACKED MESSAGE "ERROR -"
98F0	PUTMSG			; PUT MESSAGE ON BOTTOM LINE OF SCREEN & PRINTER
98F0 A0 BD		LDY	#PROMSG/256	INPUT: A=MSG LSBYTE
98F2	PTMSG2			; MSG MSB
98F2 A6 BD		LDX	DSPFLG	; INPUT A=MSG LSB, Y=MSG MSB
98F4 D0 30		BNE	SETRTN	
98F6 A2 02		LDX	#2	
98F8 86 55		STX	COLCRS	
98FA A2 17		LDX	#23	
98FC 86 54		STX	ROWCRS	

```

9BFE 20 06 9C      JSR      SETMSG      ; SET UP MSG IN TOKBUF
9C01 4C 05 A2      JMP      PTIXTP      ; PUT TOKBUF ON SCREEN & PRINTER

9C04 A0 BD          STMSG2 LDY      #PROMSG/256      ; SET UP MSG IN TOKBUF: A=MSG LSB, MSG MSB = PROMSG/256
9C06                SETMSG                                     ; SET UP MESSAGE IN TOKBUF: A=MSG LSB, Y=MSG MSB
9C06 A6 BD          LDX      DSPFLG
9C08 D0 1C          BNE      SETRTN
9C0A A2 00          LDX      #TOKBUF
9C0C 86 82          STX      TOKPTR
9C0E 85 8C          STA      PKPTR
9C10 84 8D          STY      PKPTR+1
9C12 A0 00          LDY      #0
9C14 84 80          STY      LFRT
9C16 B1 8C          LDA      (PKPTR),Y
9C18 85 8A          STA      KEYLN2
9C1A 98             TYA
9C1B 20 1F A3      JSR      UNPCK2
9C1E 85 82          STA      TOKPTR
9C20                TOKINT
9C20 A6 B2          LDX      TOKPTR      ; SUBROUTINE TO LOAD A,X,Y FOR TOKBUF DISPLAY
9C22                TKINT2
9C22 A9 00          LDA      #TOKBUF
9C24 A0 05          LDY      #TOKBUF/256
9C26                SETRTN
9C26 60             RTS

9C27 A6 B2          INVID  LDX      TOKPTR      ; CHANGE TOKBUF TO INVERSE VIDEO (EXCEPT BLANKS) & LOAD A,X,Y FOR DISPLAY
9C29                CHSLP
9C29 BD 00 05      LDA      TOKBUF,X
9C2C C9 20          CMP      #'
9C2E F0 05          BEQ      INV10      ; IF BLANK THEN NO INVERSE VIDEO
9C30 09 80          ORA      #$80      ; INVERSE VIDEO BIT
9C32 9D 00 05      STA      TOKBUF,X
9C35                INV10
9C35 CA            DEX
9C36 10 F1          BPL      CHSLP
9C38 30 E6          BMI      TOKINT      ; JMP          LOAD A,X,Y

9C3A                SOUND      ; MAKE SOUND AT FREQ A
9C3A BD 00 D2      STA      AUDF1
9C3D A9 AF          LDA      #$AF
9C3F 8D 01 D2      STA      AUDC1
9C42 A0 80          LDY      #$80      ; DELAY
9C44 AA            TAX
9C45                SNDLP1
9C45 CA            DEX
9C46 D0 FD          BNE      SNDLP1
9C48 88            DEY
9C49 D0 FA          BNE      SNDLP1
9C4B 8C 01 D2      STY      AUDC1      ; TURN SOUND OFF
9C4E 60             RTS

```


COLLEEN CALCULATOR, BY C SHAW

9C4F

CLNUM

; CLEAR TOKBUF SO NUMBER CAN BE LOADED
RETURN A=' X=\$FF Y UNCHANGED

9C4F A2 0D

LDX

#NUMLEN-1

9C51 A9 20

LDA

#'

9C53

CLNLP

9C53 9D 00 05

STA

TOKBUF, X

9C56 CA

DEX

9C57 10 FA

BPL

CLNLP

9C59 60

RTS

COLLEEN CALCULATOR, BY C SHAW

9C5A	DSPRG			; DISPLAY PROGRAM ADDRESS, CURRENT TOKEN RETURN CS=>ERROR CC=>NO ERROR
9C5A A5 BA		LDA	PC+1	
9C5C C5 D3		CMP	PC1MX1	
9C5E 90 05		BCC	DG10	
9C60	EPERR			
9C60 A9 B6		LDA	#EPMMSG	; PAST END OF MEM
9C62 4C B7 9B		JMP	ERRSUB	
9C65	DG10			
9C65 A6 BD		LDX	DSPFLG	
9C67 D0 76		BNE	DG80	; NO DISPLAY IF EXEC & NOTRACE
9C69 20 BB 9F		JSR	FPU SHO	; SAVE X
9C6C A5 BA		LDA	PC+1	; RELOAD PC MSB
9C6E 38		SEC		
9C6F E5 D1		SBC	PRGADR+1	; NORMALIZE TO 0 BASE INSTEAD OF PRGMEM
9C71 85 D5		STA	FRO+1	
9C73 A5 B9		LDA	PC	
9C75 85 D4		STA	FRO	
9C77 20 AA D9		JSR	IFP	
9C7A 20 E6 D8		JSR	FASC	; 0 -> 1023 IN (INBUFF) IN ASCII
9C7D 20 9D 9F		JSR	FPOPO	; RELOAD X
9C80 20 97 A2		JSR	PTCRPN	; PUT CR ON PRINTER IF PREVIOUS WAS #, NUMFLG<-0
9C83 A0 FF		LDY	#\$FF	; RIGHT JUSTIFY IN TOKBUF, PUT 0'S AT LEFT
9C85	DGLP1			
9C85 C8		INY		; FIND END OF BUFFER
9C86 B1 F3		LDA	(INBUFF),Y	
9C88 10 FB		BPL	DGLP1	
9C8A 29 7F		AND	#\$7F	; MASK OFF END OF BUFFER INDICATOR
9C8C A2 03		LDX	#3	
9C8E D0 02		BNE	DG20	; JMP
9C90	DGLP2			; MOVE TO TOKBUF
9C90 B1 F3		LDA	(INBUFF),Y	
9C92	DG20			
9C92 9D 00 05		STA	TOKBUF, X	
9C95 CA		DEX		
9C96 88		DEY		
9C97 10 F7		BPL	DGLP2	
9C99 BA		TXA		
9C9A 30 08		BMI	DG30	
9C9C A9 30		LDA	#0	; PAD WITH 0'S
9C9E	DGLP3			
9C9E 9D 00 05		STA	TOKBUF, X	
9CA1 CA		DEX		
9CA2 10 FA		BPL	DGLP3	
9CA4	DG30			
9CA4 20 27 A2		JSR	PUTDEL	
9CA7 E6 55		INC	COLORS	; ONE SPACE AT BEGINNING OF LINE
9CA9 20 22 9C		JSR	TRINT2	; AC-TOKBUF, YC-TOKBUF/256
9CAC A2 04		LDX	#4	
9CAE 20 AB A2		JSR	PITCHBP	
9CB1 20 61 A1		JSR	NCHALD	; LOAD TOKEN CODE FROM PRGMEM, CHECK FOR NUMBER
9CB4 80 78		BCC	DGRTN	; ERROR
9CB6 D0 06		BNE	DG6Q	; NOT NUMBER
9CB8	DG40			
9CB8 20 B6 9D		JSR	TOKNUM	; IF NUMBER -> ASCII IN TOKBUF
9CB8 4C D4 9C		JMP	DG70	

9CB0 20 88 7D
9CBB 4C D4 9C

JSR
JMP

TOKNUM
DG70

;FP NUMBER -> ASCII IN TOKBUF

COLLEEN CALCULATOR, BY C SHAW

9CBE DG60
9CBE 85 B1 STA TOKCOD ;NOT A #
9CC0 PUTCMD ;ENTRY POINT TO DISPLAY COMMAND
9CC0 20 47 A3 JSR UNPKEY ;UNPACK CHARS FOR TOKEN
9CC3 B0 1C BCS DGRTN ;ERROR
9CC5 A9 0E LDA #NUMLEN
9CC7 38 SEC
9CC8 E5 82 SBC TOKPTR ;Y = LENGTH OF BUFFER
9CCA F0 08 BEQ DG70
9CCC AA TAX ;OUTPUT BLANKS
9CCD A9 28 LDA #BLKBUF
9CCF A0 05 LDY #BLKBUF/256
9CD1 20 AB A2 JSR PTCHSP
9CD4 DG70
9CD4 E6 55 INC COLCRS ;ONE COLUMN TO RIGHT
9CD6 20 20 9C JSR TOKINT ;SET UP A=TOKBUF, X=TOKPTR, Y=TOKBUF/256
9CD9 20 AB A2 JSR PTCHSP
9CDC 20 0B A2 JSR PUTCR ;CR ON SCREEN ONLY, NOT PRINTER
9CDF DG80
9CDF 18 CLC
9CE0 60 RTS

9CE1 DGRTN
9CE1 20 0B A2 JSR PUTCR ;PUT EXTRA LINE AFTER ERROR MSG
9CE4 38 SEC
9CE5 60 RTS

9CE6	FPBIN			; CONVERT FRO TO 32 BIT BINARY #
				THEN COMPARE WITH BITBIN TO SEE IF IT IS IN THE
				RANGE SPECIFIED BY BITINT.
9CE6 A5 D4		LDA	FRO	
9CEB 85 A0		STA	NEGFLG	
9CEA 10 04		BPL	FP10	
9CEC 29 7F		AND	#\$7F	; TAKE ABSOLUTE VALUE
9CEE 85 D4		STA	FRO	
9CF0	FP10			
9CF0 20 2E 9D		JSR	FPBNCK	; CONVERT FP TO BINARY
9CF3 B0 61		BCS	FP21	; OVERFLOW
9CF5 A5 A0		LDA	NEGFLG	
9CF7 10 03		BPL	FP20	
9CF9 20 06 A4		JSR	S2CMP	; NEG # - TAKE COMP, ADD 1
9CFC	FP20			
9CFC	BINCHK			; CHECK TO SEE IF BINARY IS WITHIN RANGE
				AS SPECIFIED BY CURRENT BITS.
				CS => ERROR CC => NO ERROR
9CFC A5 B0		LDA	BINARY	
9CFE 30 11		BMI	BINC10	
9D00 A2 00		LDX	#0	; POSITIVE
9D02	BINLP1			
9D02 B5 B0		LDA	BINARY, X	
9D04 D5 A4		CMP	BITBIN, X	
9D06 F0 02		BEQ	BINOK	
9D08 B0 16		BCS	BOERR	; TOO LARGE
9D0A	BINOK			
9D0A E8		INX		
9D0B E0 04		CPX	#4	
9D0D D0 F3		BNE	BINLP1	
9D0F 18		CLC		
9D10 60		RTS		; OK
9D11	BINC10			
9D11 A2 00		LDX	#0	
9D13	BINLP2			
9D13 B5 B0		LDA	BINARY, X	
9D15 D5 AC		CMP	BINMIN, X	
9D17 90 07		BCC	BOERR	; TOO SMALL
9D19 E8		INX		
9D1A E0 04		CPX	#4	
9D1C D0 F5		BNE	BINLP2	
9D1E 18		CLC		
9D1F 60		RTS		; OK
9D20	BOERR			
9D20 A2 03		LDX	#3	; CLEAR BINARY BECAUSE OF ERROR
9D22 A9 00		LDA	#0	
9D24 95 B0	BINLP3	STA	BINARY, X	
9D26 CA		DEX		
9D27 10 FB		BPL	BINLP3	
9D29 A9 5C		LDA	#BOMSG	; BINARY UNDERFLOW
9D2B 4C B7 9B		JMP	ERRSUB	

9D2E

FPBNCK

; CONVERT POSITIVE FRO TO 4 BYTE BINARY
; CLEAR BINARY IF TOO LARGE AND SET CARRY

9D2E 20 BB 9F

JSR

FPUSHO

; SAVE COPY OF # FOR MOD

9D31 A2 30

LDX

#C65536

9D33 A0 BA

LDY

#C65536/256

9D35 20 89 DD

JSR

FLDOR

9D38 20 E8 A6

JSR

SMOD

; FRO = X MOD 65536 (2 LOWER BYTES)

9D3B 20 D2 D9

JSR

FPI

; MODFAC = INT(X/65536) (2 UPPER BYTES)

9D3E A5 D4

LDA

FRO

; LSB

9D40 85 B3

STA

BINARY+3

9D42 A5 D5

LDA

FRO+1

9D44 85 B2

STA

BINARY+2

9D46 20 12 9F

JSR

FLDOM

; LOAD MODFAC

9D49 20 D2 D9

JSR

FPI

9D4C 80 D2

BCS

BOERR

; ERROR: OVERFLOW

9D4E A5 D4

LDA

FRO

9D50 85 B1

STA

BINARY+1

9D52 A5 D5

LDA

FRO+1

; MSB

9D54 85 B0

STA

BINARY+0

9D56 60

FP21

RTS

```

9D57          FDSP0
;
9D57 A5 BD    LDA    DSPFLG    CONVERT FRO TO ASCII AND DISPLAY
9D59 D0 1F    BNE    DSPRTN    ; RETURN IF NO DISPLAY
9D5B 20 27 A2 JSR    PUTDEL    ; CLEAR BOTTOM LINE & SET UP CURSOR
9D5E 20 86 9D JSR    TOKNUM    ; CONVERT FRO TO ASCII IN TOKBUF
; DAYDSP
9D61 20 97 A2 JSR    PTCRPN    ; DISPLAY TOKBUF ON SCREEN AND PRINTER (IF ON)
9D64 A9 04    LDA    #4        ; PUT CR ON PRINTER IF PREVIOUS THING WAS NUMBER
9D66 20 6F 9F JSR    PUTBLK    ; PUT 4 BLANKS ON PRINTER ONLY
9D69 A9 07    LDA    #7
9D6B 85 55    STA    COLCRS
9D6D 20 20 9C JSR    TOKINT    ; SET UP A=TOKBUF, X=TOKPTR, Y=TOKBUF/256
9D70 20 AB A2 JSR    PTCHSP
9D73 20 0B A2 JSR    PUTCR
9D76 A9 01    LDA    #1
9D78 85 A2    STA    NUMFLG
9D7A 60       DSPRTN RTS

```

```

9D7B          FDSP1
9D7B 85 55    STA    COLCRS    ; DISPLAY NUMBER ON SCREEN ONLY, IN COL A
9D7D 20 86 9D JSR    TOKNUM
9D80          FDSP2
9D80 20 20 9C JSR    TOKINT    ; DISPLAY TOKBUF # ON SCREEN WHEREVER CURSOR IS
9D83 4C 74 A2 JMP    PUTCHS    ; SET UP A=TOKBUF, X=TOKPTR, Y=TOKBUF/256

```


COLLEEN CALCULATOR, BY C SHAW

```

9DB6          TOKNUM          ; CONVERT FRO TO ASCII IN TOKBUF -> TOKBUF+NUMLEN-1 (RIGHT JUSTIFIED)
9DB6 A5 BD          LDA      DSPFLG
9DB8 D0 F0          BNE      DSPRTN          ; RETURN IF NO DISPLAY
9DBA 20 BB 9F          JSR      FPUSH0          ; SAVE FP #
9DBD A5 B7          LDA      DHOFLG
9DBF F0 49          BEQ      FDS05

```

```

          NON-DECIMAL NUMBER
9D91 20 7D A6          JSR      STRUNC          ; TRUNCATE # FOR DISPLAY ONLY
9D94 20 E6 9C          JSR      FPBIN
9D97 90 0A          BCC      TOKN10

          ; OVERFLOW ERROR => DISPLAY MSG INSTEAD OF NUMBER
          ; "HEX/OCT OVRFLW"
9D99 A9 5C          LDA      #BOMSG
9D9B A0 BC          LDY      #ERRTBL/256
9D9D 20 06 9C          JSR      SETMSG          ; TOKBUF = MESSAGE
9DA0 4C 0B 9F          JMP      FABCD
9DA3          TOKN10

```

```

9DA3 A2 03          LDX      #3

```

```

          LIMIT TO # OF BITS SET BY "BITS" COMMAND
9DA5          FDLP3
9DA5 B5 B0          LDA      BINARY,X
9DA7 35 AB          AND      BITBN2,X
9DA9 95 B0          STA      BINARY,X
9DAB CA          DEX
9DAC 10 F7          BPL      FDLP3

```

```

9DAE A9 00          LDA      #0
9DB0 20 27 AB          JSR      BINFP2          ; TREAT ALL AS POSITIVE

9DB3 20 4F 9C          JSR      CLNUM          ; CLEAR TOKBUF
9DB6 A9 0E          LDA      #NUMLEN
9DB8 B5 B2          STA      TOKPTR

```

```

9DBA          FDLP2
9DBA A5 B7          LDA      DHOFLG
9DBC 20 E0 A6          JSR      INTMOD          ; FRO <- FRO MOD DHOFLG
9DBF A5 D5          LDA      FRO+1          ; 0-> (DHOFLG-1) IN BCD
9DC1 18          CLC
9DC2 69 30          ADC      #'0          ; CONVERT TO ASCII
9DC4 C9 40          CMP      #'0+$10          ; 10 TO 15?
9DC6 90 02          BCC      FDS1          ; NO.
9DC8 69 00          ADC      #'A-'0-$10-1          ; YES. CONVERT TO ASCII A TO F
9DCA          FDS1
9DCA A0 00          LDY      #0
9DCC C6 B2          DEC      TOKPTR
9DCE 91 B2          STA      (TOKPTR),Y

```

```

9DD0 20 12 9F          JSR      FLDOM          ; LOAD MODFAC=INT(FRO/DHOFLG)
9DD3 A5 D4          LDA      FRO
9DD5 D0 E3          BNE      FDLP2
9DD7 4C 0B 9F          JMP      FABCD

```

9DDA	FDB05	STA	BEFORM	DECIMAL MODE	A = DHOFLG = 0
9DDA 85 BF		JER	CLNUM	INIT TO 0 TO INDICATE NOT EFORM	
9DDC 20 4F 9C				CLEAR TOKBUF RETURN A=' X=\$FF Y UNCHANGED	
9DDF A2 50		LDX	#\$50	ADD 5X10^K IN PREPARATION FOR ROUNDING	
9DE1 A5 D5		LDA	FRO+1		
9DE3 85 C3		STA	SMSD	SAVE MSD OF NUMBER	
9DE5 A5 D4		LDA	FRO		
9DE7 85 C2		STA	SSIGN	SAVE SIGN	
9DE9 F0 6A		BEG	RF90	10 => NOTHING FANCY NEEDED	
9DEB 29 7F		AND	#\$7F	TAKE ABSVAL	
9DED 85 D4		STA	FRO		
9DEF C9 3F		CMP	#\$3F	E FORM?	
9DF1 90 14		BCC	RF10	YES.	
9DF3 C9 44		CMP	#\$44		
9DF5 B0 10		BCC	RF10	YES.	
9DF7 A5 C0		LDA	FIXNUM	NO.	
9DF9 C9 08		CMP	#8	FIX 8-(NOFIX)?	
9DFB F0 12		BEG	RF20	YES.	
9DFD 4A		LSR	A	DIVIDE BY 2	
9DFE 90 02		BCC	RF70	ODD?	
9E00 A2 05		LDX	#\$05	YES.	
9E02	RF70				
9E02 49 3F		EOR	#\$3F	TAKE COMPLEMENT, CHANGE TO EXCESS 40 NOTATION	
9E04 4C 29 9E		JMP	RF80		
9E07	RF10				
9E07 E6 BF		INC	SEFORM	YES. SET FLAG	
9E09 A5 C0		LDA	FIXNUM	FORM 5X10^(-(FIXNUM+1))X10^K1	
9E0B C9 08		CMP	#7+1	WHERE K1=POWER OF 10 IN FRO.	
9E0D 90 02		BCC	RF30	MAX # OF DIGITS AFTER DP=7 IN E MODE	
9E0F	RF20				
9E0F A9 07		LDA	#7		
9E11	RF30				
9E11 A4 D5		LDY	FRO+1		
9E13 C0 10		CPY	#\$10	IN RANGE 0-9?	
9E15 90 04		BCC	RF40	YES.	
9E17 E9 01		SBC	#1	NO. 10-99 => ODD POWER OF 10, 5X10^(-(FIXNUM-1)+1)X10^(K1-1)	
9E19 C9 FF		CMP	#\$FF	SET CARRY IF NEG. TO BE ROTATED INTO MSD	
9E1B	RF40				
9E1B 6A		ROR	A	DIVIDE BY 2 TO GET POWER OF 100	
9E1C 90 02		BCC	RF50	ODD POWER OF 10?	
9E1E A2 05		LDX	#\$05	YES. USE 5, NOT \$50	
9E20	RF50				
9E20 49 FF		EOR	#\$FF	TAKE COMPLEMENT = -(A-1)	
9E22 18		CLC			
9E23 65 D4		ADC	FRO	COMBINE WITH EXPONENT	
9E25 C9 10		CMP	#\$F+1	TOO SMALL?	
9E27 90 3A		BCC	RFERR	YES. ERROR	
9E29	RF80				
9E29 85 E0		STA	FR1	EXPONENT	
9E2B 86 E1		STX	FR1+1	5 OR \$50	
9E2D A2 03		LDX	#3		
9E2F A9 00		LDA	#0		
9E31	RFLP1				
9E31 85 E2		STA	FR1+2	CLEAR REST OF FR1	
9E33 CA		DEX			

COLLEEN CALCULATOR, BY C SHAW

```

9E34 10 FB          BPL      RFLP1
9E36 20 66 DA      JSR      FADD      ; FINALLY CAN DO ADD
9E39 B0 28          BCS      RFERR    ; ERROR - OVERFLOW

9E3B A5 BF          LDA      SEFORM    ; EFORM?
9E3D D0 08          BNE      RF85     ; YES.
9E3F A5 D4          LDA      FRO      ; NO.
9E41 C9 44          CMP      ##44     ; EXPONENT TOO LARGE?
9E43 90 10          BCC      RF90     ; NO. OK
9E45 E6 BF          INC      SEFORM    ; YES.  EFORM AFTER ALL  MUST HAVE BEEN APPROX 99999999.9 TO 100000000
9E47              RF85
9E47 20 51 DA      JSR      INTLBF    ; YES. MAKE INBUFF POINT TO LBUFF
9E4A A9 30          LDA      #'0      ; STORE ASCII 0
9E4C 8D 75 05      STA      LBUFF-11
9E4F 20 20 D9      JSR      XEFORM    ; FP -> E FORM ASCII
9E52 4C 58 9E      JMP      RF100
9E55              RF90
9E55 20 E6 D8      JSR      FASC      ; FP -> ASCII (NOT EFORM WE HOPE)
9E58              RF100

;
9E58 A0 FF          LDY      ##FF      FIND & SAVE E+/-NN
9E5A              RFLP2
9E5A C8            INY
9E5B B1 F3          LDA      (INBUFF),Y
9E5D 10 16          BPL      RF110
9E5F A6 BF          LDX      SEFORM    ; END OF BUFFER
9E61 F0 09          BEQ      RF105
9E63              RFERR
9E63 20 88 A9      JSR      CRYSDND   ; ERROR: SHOULD HAVE E
9E66 20 86 9F      JSR      FPOP1     ; "ARITHMETIC OVERFLOW" ERROR - CLEAR FRO
9E69 4C 86 9D      JMP      TOKNUM    ; POP OFF OLD X VALUE
; TRY DISPLAY AGAIN

9E6C              RF105
9E6C 29 7F          AND      ##7F     ; CLEAR END OF BUFFER INDICATOR
9E6E 91 F3          STA      (INBUFF),Y
9E70 84 C1          STY      MANTLN
9E72 4C 91 9E      JMP      RF120    ; SAVE MANTISSA LENGTH

9E75              RF110
9E75 C9 45          CMP      #'E      ; E FOUND?
9E77 D0 E1          BNE      RFLP2    ; NO. CONTINUE

9E79 A6 BF          LDX      SEFORM    ; YES. EFORM?
9E7B F0 E6          BEQ      RFERR    ; NO. ERROR, SHOULD NOT HAVE E
9E7D 88            DEY
9E7E 84 C1          STY      MANTLN    ; SAVE MANTISSA LENGTH (ADDR OF LAST CHAR)
9E80 C8            INY
9E81 A2 0A          LDX      #NUMLEN-4 ; MOVE E TO TOKBUF
9E83 9D 00 05      STA      TOKBUF,X
9E86 E8            INX
9E87 C8            INY
9E88 B1 F3          LDA      (INBUFF),Y
9E8A 10 F7          BPL      RFLP3
9E8C 29 7F          AND      ##7F
9E8E 9D 00 05      STA      TOKBUF,X
9E91              RF120

```

NOW FIND DECIMAL PT

9E91 A0 FF		LDY	#0FF	
9E93	RFLP4			
9E93 C8		INY		
9E94 C4 C1		CPY	MANTLN	
9E94 F0 0A		BEG	RF130	
9E98 90 08		BCC	RF130	; NOT AT END YET
9E9A A1 2E		LDA	#'	; ADD ' '
9E9C 91 F3		STA	(INBUFF),Y	
9E9E 84 C1		STY	MANTLN	
9EA0 D0 06		BNE	RF140	; JMP
9EA2	RF130			
9EA2 B1 F3		LDA	(INBUFF),Y	
9EA4 C9 2E		CMP	#'	
9EA4 D0 EB		BNE	RFLP4	
9EA8	RF140			; HAVE ' '
9EA8 A6 C0		LDX	FIXNUM	
9EAA E0 08		CPX	#8	
9EAC D0 20		BNE	RF148	; FIXED DEC
9EAE A4 C1		LDY	MANTLN	; NOT FIXED DEC
9EB0 C0 09		CPY	#9	
9EB2 90 3D		BCC	RF170	; <= 8 DIGITS IS OK
9EB4 A0 08		LDY	#8	
9EB6 A5 C2		LDA	SSIGN	; LOAD OLD FRO VALUE
9EB8 29 7F		AND	#\$7F	
9EBA C9 3F		CMP	#\$3F	; NON EFORM FRACTION (E.G. 1, .01)?
9EBC D0 07		BNE	RF142	; NO.
9EBE	INY			; YES. ALLOW EXTRA DIGIT FOR LEADING '0'
9EBE A5 C3		LDA	SMSD	; LOAD OLD FRO+1
9EC0 C9 10		CMP	#\$10	; 1 OR 2 DIGITS?
9EC2 80 01		BCS	RF142	; 2
9EC4 C8		INY		; 1 => ALLOW EXTRA CHAR FOR 0 AFTER D.P.
9EC5	RF142			
9EC5 B1 F3		LDA	(INBUFF),Y	
9EC7 C9 30		CMP	#'0	
9EC9 D0 26		BNE	RF170	
9ECB 88		DEY		
9ECC 10 F7		BPL	RF142	; JMP
9ECE	RF148			
9ECE 98		TYA		
9ECF 18		CLC		; COMPUTE WHERE END OF NUMBER SHOULD BE
9ED0 65 C0		ADC	FIXNUM	
9ED2 C5 C1		CMP	MANTLN	
9ED4 B0 04		BCS	RF150	
9ED6 85 C1		STA	MANTLN	; MANTISSA TOO LONG => DISCARD DIGITS
9ED8 90 0F		BCC	RF160	; JMP
9EDA F0 0D	RF150	BEG	RF160	; JUST RIGHT
9EDC A4 C1		LDY	MANTLN	; MANTISSA TOO SHORT: PAD WITH 0'S
9EDE 85 C1		STA	MANTLN	; NEW MANTISSA LENGTH
9EE0 A9 30		LDA	#'0	
9EE2	RFLP5			
9EE2 C8		INY		
9EE3 91 F3		STA	(INBUFF),Y	
9EE5 C4 C1		CPY	MANTLN	; REACHED DESIRED LENGTH?
9EE7 D0 F9		BNE	RFLP5	; NO. CONTINUE
9EE9	RF160			LIMIT TO 8 DIGITS MAX + DP
9EE9 A4 C1		LDY	MANTLN	
9EEB C0 09		CPY	#9	

9EE9		RF160	LDY	MANTLN
9EE9 A4 C1			CPY	#9
9EEB C0 09				

COLLEEN CALCULATOR, BY E. SHAW

9EED 90 02		BCC	RF170	;OK
9EEF A0 08		LDY	#8	
9EF1	RF170			;MOVE TO TOKBUF
9EF1 A2 0D		LDX	#NUMLEN-1	
9EF3 A3 BF		LDA	SEFORM	;E FORM?
9EF5 F0 02		BEG	FDLP4	;NO.
9EF7 A2 09		LDX	#NUMLEN-5	;YES. ALLOW ROOM FOR EXPONENT
9EF9	FDLP4			
9EF9 B1 F3		LDA	(INBUFF),Y	
9EFB 9D 00 05		STA	TOKBUF,X	
9EFE CA		DEX		
9EFF 88		DEY		
9F00 10 F7		BPL	FDLP4	
9F02 A5 C2		LDA	SSIGN	CHECK SIGN
9F04 10 05		BPL	FABCD	
9F06 A9 2D		LDA	#'-	;NEGATIVE => STORE '-'
9F08 9D 00 05		STA	TOKBUF,X	
9F08	FABCD			
9F08 A9 0E		LDA	#NUMLEN	
9F0D 85 B2		STA	TOKPTR	
9F0F 4C 9D 9F		JMP	FPOPO	;POP ORIGINAL # OFF STACK

```

9F12          FLDOM          LDX      #MODFAC          ; FRO <- MODFAC
9F12 A2 50
9F14 A0 05          LDY      #MODFAC/256
9F16 D0 35          BNE      CFLD02
9F18          FLDOS          JSR      FPOPLD          ; FRO <- TOP OF STACK
9F18 20 A9 9F          BNE      CFLD02          ; LOAD X&Y REGS WITH STACK POINTER
9F18 D0 30          ; JMP

9F1D          SRECTA        ; ->POLAR NEW X=R*SQRT(SQU(X)+SQU(Y)) NEW Y=THETA=ASIN(Y/R)
9F1D A9 61          LDA      #ZRECT
9F1F 20 F0 98          JSR      PUTMSG          ; DISPLAY "-> POLAR"
9F22 20 5A A8          JSR      SSQUAR          ; X*X
9F25 20 55 9F          JSR      FSTOT
9F28 20 18 9F          JSR      FLDOS
9F2B 20 5A A8          JSR      SSQUAR          ; Y*Y
9F2E 20 4F 9F          JSR      FLD1T
9F31 20 6A A9          JSR      SFADD
9F34 20 A7 B1          JSR      SSQRT          ; R = NEW X (TOS)
9F37 20 B6 DD          JSR      FMOVE
9F3A 20 55 9F          JSR      FSTOT
9F3D 20 9D 9F          JSR      FPOPO          ; Y
9F40 20 3A A9          JSR      SFDIV          ; Y/R
9F43 20 05 B1          JSR      SASIN          ; THETA = NEW Y
9F46 20 BB 9F          JSR      FPUSHO
9F49          FLDOT          ; FRO <- FTEMP
9F49 A2 56          LDX      #FTEMP
9F4B A0 05          LDY      #FTEMP/256
9F4D          CFLD02          BNE      CFLDOR          ; JMP
9F4D D0 51

9F4F          FLD1T          ; FR1 <- FTEMP
9F4F A2 56          LDX      #FTEMP
9F51 A0 05          LDY      #FTEMP/256
9F53 D0 34          BNE      CFLD1R          ; JMP
9F53 D0 34

9F55          FSTOT          ; FTEMP <- FRO
9F55 A2 56          LDX      #FTEMP
9F57 A0 05          LDY      #FTEMP/256
9F59 4C A7 DD          JMP      FSTOR
9F5C          FST1T          ; FTEMP <- FR1
9F5C A2 56          LDX      #FTEMP
9F5E A0 05          LDY      #FTEMP/256
9F60          FST1R          ; (X,Y) <- FR1
9F60 B6 FC          STX      FLPTR
9F62 84 FD          STY      FLPTR+1
9F64 A0 05          LDY      #5
9F66          FSLOP          LDA      FR1,Y
9F66 B9 E0 00          STA      (FLPTR),Y
9F69 91 FC          DEY
9F6B 88          BPL      FSLOP
9F6C 10 F8          PUTBRT
9F6E          PUTBLK          RTS
9F6E 60

9F6F          PUTBLK          LDX      PRNFLG          ; PUT A BLANKS ON PRINTER ONLY
9F6F A6 95          BEQ      PUTBRT
9F71 F0 FB          LDX      #PIOCB
9F73 A2 20          STA      ICBLL,X
9F75 9D 4B 03

```


9FD0 A9 04		LDA	#4	; N
9FD2 20 B8 A3		JSR	MEMLD1	
9FD5 20 EB 9F		JSR	FPUSH1	; SAVE N FOR LATER
9FD8 20 3A A9		JSR	SFDIV	
9FD8 20 B6 DD		JSR	FMOVE	
9FDE 68		PLA		; RELOAD REG #
9FDF 18		CLC		; AND INCREMENT
9FE0 69 01		ADC	#1	; SIGMA(SQU)
9FE2 20 B2 A3		JSR	MEMLDO	
9FE5 20 83 A9		JSR	SFSUB	
9FEB	SXCHGY			; X<==>Y FRO<==>TOS
9FEB 20 9A 9F		JSR	FMVPOP	
9FEB	FPUSH1			
9FEB 20 E3 9F		JSR	FPSHLD	; PUSH FR1 ON FPSTK
9FEE 20 60 9F		JSR	FST1R	
9FF1 30 CE		BMI	FPSH05	; JMP
9FF3	FPSHLD			
9FF3 A6 9B		LDX	FPPTR	; LOAD REGISTERS & CHECK FOR OVERFLOW
9FF5 E0 FC		CPX	#FPSLEN*FPREC	
9FF7 90 09		BCC	FPC10	
9FF9 A9 74		LDA	#NSFMSG	
9FFB 20 B7 9B		JSR	ERRSUB	; STACK OVERFLOW
9FFE A2 F6		LDX	#FPSLEN-1*FPREC	
A000 86 9B		STX	FPPTR	
A002	FPC10			
A002 A0 06		LDY	#FPSTK/256	
A004 60		RTS		

A005	SUBCAL			;PERFORM OP A (ROUTINE CALLED WILL DO RTS)
A005 C9 8D		CMP	#EQUAL+1	
A007 90 03		BCC	SBCL5	
A009 4C B5 9B		JMP	KEYERR	
A00C	SBCL5			
A00C 0A		ASL	A	
A00D AB		TAY		
A00E A9 13		LDA	#JMPTBL	
A010 85 90		STA	JMPTR1	
A012 A9 BB		LDA	#JMPTBL/256	
A014 85 91		STA	JMPTR1+1	
A016 90 02		BCC	MAIN10	
A018 E6 91		INC	JMPTR1+1	
A01A	MAIN10			;SECOND PAGE OF TABLE
A01A B1 90		LDA	(JMPTR1),Y	
A01C 85 92		STA	JMPTR2	
A01E CB		INY		;LOAD AND STORE JSR ADDRESS
A01F B1 90		LDA	(JMPTR1),Y	
A021 85 93		STA	JMPTR2+1	
A023 6C 92 00		JMP	(JMPTR2)	

A026	GTCHR			RETURN NEXT INPUT CHAR IN A REG
A026 A6 86		LDX	TOKTIN	
A028 F0 07		BEQ	GETC05	
A02A C6 86		DEC	TOKTIN	
A02C B5 83		LDA	TOKTMP-1, X	;USE CHARS FROM PREVIOUS CALL
A02E 4C 81 A0		JMP	GETC10	;SAVE CHAR IN TOKBUF
A031	GETC05			
A031 A2 10		LDX	#KIOCB	
A033 A9 07		LDA	#GTCHR	;GET FROM K: (DATA RETURNED IN A, STATUS IN Y)
A035 9D 42 03		STA	ICCOM, X	
A038 A5 82		LDA	TOKPTR	
A03A 9D 44 03		STA	ICBAL, X	
A03D A5 83		LDA	TOKPTR+1	
A03F 9D 45 03		STA	ICBAH, X	
A042 A9 01		LDA	#1	
A044 9D 48 03		STA	ICBL, X	
A047 A9 00		LDA	#0	
A049 9D 49 03		STA	ICBLH, X	
A04C 20 56 E4		JSR	CIOV	
A04F C0 01		CPY	#SUCCES	
A051 D0 36		BNE	GETC12	;BREAK => DELETE LINE
A053 C9 9B		CMP	##9B	
A055 90 14		BCC	GETC06	;0-9A
A057 D0 04		BNE	GNOCR	
A059 A9 20		LDA	#'	;9B
A05B D0 3C		BNE	GETC30	;CR => CHANGE TO BLANK AND DON'T PRINT
A05D	GNOCR			
A05D C9 FD		CMP	##FD	;FD=BELL
A05F B0 2C		BCS	GETC15	;FD=FF
A061 C9 A0		CMP	##A0	

COLLEEN CALCULATOR, BY C SHAW

A063 B0 06	BCS	GETC06	; A0-FD
A065 C9 9C	CMP	#DELLIN	; 9C-9F
A067 F0 2B	BEG	GETC20	; DON'T ESCAPE IF DELETE LINE (9C)
A069 D0 22	BNE	GETC15	; 9D-9F
A06B	GETC06		; 0-9A OR A0-FD
A06B 29 7F	AND	##7F	; STRIP OFF INVERSE VIDEO, IF ANY
A06D F0 1E	BEG	GETC15	; 0
A06F C9 1B	CMP	##1B	
A071 B0 04	BCS	GETC07	
A073 69 40	ADC	#2*32	; 1-1A (CONVERT CTRL GRAPHICS TO UPPER CASE LETTER)
A075 D0 0A	BNE	GETC10	; JMP
A077	GETC07		
A077 C9 61	CMP	#'A+32	; LOWER CASE ALPHA TO UPPER CASE
A079 90 06	BCC	GETC10	
A07B C9 7B	CMP	#'Z+32+1	
A07D B0 02	BCS	GETC10	
A07F E9 1F	SBC	#32-1	; CARRY SET
A081	GETC10		
A081 C9 20	CMP	#'	
A083 F0 14	BEG	GETC30	; DON'T PRINT SPACE
A085 C9 7E	CMP	#BACKSP	
A087 D0 04	BNE	GETC15	
A089	GETC12		
A089 A9 9C	LDA	#DELLIN	
A08B D0 07	BNE	GETC20	; JMP BACKSPACE IS EQUIV TO DELETE LINE
A08D	GETC15		
A08D 4B	PHA		
A08E A9 1B	LDA	#ESC	
A090 20 31 A2	JSR	PTCHR	
A093 6B	PLA		
A094	GETC20		
A094 4B	PHA		
A095 20 31 A2	JSR	PTCHR	; PUT ON SCREEN
A09B 6B	PLA		
A099	GETC30		
A099 EA	NOP		; TEMP SO NO CARTRIDGE B
A09A EA	NOP		
A09B EA	NOP		
A09C EA	NOP		
A09D EA	NOP		
A09E EA	NOP		
A09F EA	NOP		
A0A0 A0 00	LDY	#0	
A0A2 91 82	STA	(TOKPTR), Y	
A0A4 60	RTS		

AOA5

GETDHO

GET DEC, HEX, OR OCT DIGIT AND RETURN IN A, TOKBUF
 RETURN CC=>NO ERROR, CS=> ERROR

AOA5 20 26 A0
 AOA8 C9 9C
 AOA9 D0 05
 AOAC 68
 AOAD 68
 AOAE 4C 51 9A

JSR
 CMP
 BNE
 PLA
 PLA
 JMP

GTCHR
 #DELLIN
 DHOCHK

 LEX

; IF DELETE LINE THEN POP STACK (SKIP RETURN)

AOB1

DHOCHK

; ENTRY POINT IF ALREADY HAVE CHAR

RETURN CC=>NO ERROR, CS=> ERROR

AOB1 C9 30
 AOB3 90 0A
 AOB5 A6 87
 AOB7 E0 08
 AOB9 D0 06

CMP
 BCC
 LDY
 CPX
 BNE

#'0
 DHOERR
 DHOFLG
 #8
 DHO10

; ERROR

AOBB C9 3B
 AOB8 90 12
 AOB9
 AOB9 3B
 AOCC 60

DHOERR

CMP
 BCC

 SEC
 RTS

#'7+1
 DHOOK

; OCTAL
 ; OK OCT 0-7

; ERROR CARRY SET

AOC1
 AOC1 C9 3A
 AOC3 90 0C
 AOC5 E0 10
 AOC7 D0 F6
 AOC9 C9 41
 AOCB 90 F2
 AOCB C9 47
 AOCF B0 EE
 AOD1
 AOD1 E6 82
 AOD3 60

DHO10

CMP
 BCC
 CPX
 BNE
 CMP
 BCC
 CMP
 BCS

 INC
 RTS

#'9+1
 DHOOK
 #16
 DHOERR
 #'A
 DHOERR
 #'F+1
 DHOERR

; OK DEC OR HEX 0-9

; ERROR DEC >9

; SAVE CHAR

A0D4

GETINT

; GET INTEGER FROM 0-255 FROM KEYBOARD

USEFUL FOR MEM REG #, FIX, BITS.
 RETURN CC=> OK, CS => NOT OK
 ; SAVE FRO

A0D4 20 88 9F

JSR

FPUSHO

A0D7 A5 87

LDA

DHOFLG

A0D9 48

PHA

; SAVE DHOFLG

A0DA A9 00

LDA

#0

; FORCE DECIMAL MODE

A0DC 85 87

STA

DHOFLG

A0DE 20 51 9A

JSR

LEX

A0E1 A5 81

LDA

TOKCOD

A0E3 C9 8E

CMP

#NUMBER

A0E5 F0 08

BEQ

GI05

A0E7 68

PLA

A0E8 85 87

STA

DHOFLG

A0EA 20 9D 9F

JSR

FPOPO

; RELOAD FRO

A0ED 38

SEC

A0EE 60

RTS

A0EF

GI05

A0EF 68

PLA

A0F0 85 87

STA

DHOFLG

; RESTORE

A0F2

GINT2

; ENTRY PT. IF ALREADY HAVE FRO (MUST HAVE FRO ON STACK)

A0F2 20 D2 D9

JSR

FPI

A0F5 B0 05

BCS

GI20

A0F7 A5 D5

LDA

FRO+1

A0F9 F0 01

BEQ

GI20

A0FB 38

SEC

; ERROR

A0FC

GI20

A0FC 08

PHP

A0FD A5 D4

LDA

FRO

A0FF 48

PHA

A100 20 9D 9F

JSR

FPOPO

; RELOAD FRO

A103 68

PLA

; OLD FRO = INTEGER 0-255

A104 28

PLP

; CC OR CS

A105 60

RTS

A106

GETPRI

; INPUT: A=TOKEN CODE. OUTPUT: A=PRIORITY

A106 4A	LSR	A
A107 AA	TAX	
A108 BD 64 BF	LDA	PRIOTB, X
A108 B0 04	BCS	GPR10
A10D 4A	LSR	A
A10E 4A	LSR	A
A10F 4A	LSR	A
A110 4A	LSR	A
A111	GPR10	
A111 29 0F	AND	##F
A113 A6 94	LDX	RPNALG
A115 E0 02	CPX	#ALGNOP
A117 D0 0A	BNE	GPR20
A119 C9 0D	CMP	#PHIGH
A11B B0 06	BCS	GPR20
A11D C9 06	CMP	#POR+1
A11F 90 02	BCC	GPR20
A121 A9 05	LDA	#POR
A123	GPR20	
A123 60	RTS	

A124

LDCHR

```

A124 20 4C A1      JSR      LDNIB      RETURN A=PACKED CHAR
A127 D0 1E          BNE      LDCH10
A129 20 4C A1      JSR      LDNIB
A12C C6 8A          DEC      KEYLN2      ; 1 EXTRA BYTE
A12E C9 0F          CMP      #15         ; SPECIAL 4-NIBBLE CHAR?
A130 D0 13          BNE      LDCH05      ; NO. 2 NIBBLE CHAR
A132 20 4C A1      JSR      LDNIB      ; YES. LOAD 2 NIBBLES OF ASCII
A135 0A            ASL      A
A136 0A            ASL      A
A137 0A            ASL      A
A138 0A            ASL      A
A139 85 C4          STA      LDCAV
A13B 20 4C A1      JSR      LDNIB
A13E 05 C4          ORA      LDCAV      ; COMBINE 2 NIBBLES
A140 C6 8A          DEC      KEYLN2      ; 2 MORE EXTRA BYTES
A142 C6 8A          DEC      KEYLN2
A144 60            RTS
A145                LDCH05
;                  CLC
A145 69 10          ADC      #16
A147                LDCH10
A147 AA            TAX
A148 BD 26 BC      LDA      TABLE-1, X
A148 60            RTS

```

A14C

LDNIB

```

;                  LOAD PACKED NIBBLE FROM PKPTR+Y, LFRT
A14C A5 80          LDA      LFRT
A14E 49 01          EOR      #1
A150 85 80          STA      LFRT
A152 F0 08          BEQ      LDN20
A154 CB            INY
A155 B1 BC          LDA      (PKPTR), Y
A157 4A            LSR      A
A158 4A            LSR      A
A159 4A            LSR      A
A15A 4A            LSR      A
A15B 60            RTS
A15C                LDN20
A15C B1 BC          LDA      (PKPTR), Y      ; RIGHT NIBBLE
A15E 29 0F          AND      #$F
A160 60            RTS

```


COLLEEN CALCULATOR, BY C SHAW

A161	NCHKLD			
; IF TOKEN IS NUMBER THEN LOAD NUMBER INTO FRO FROM PRGMEM				
RETURN EQ => NUMBER, NE => NOT #, CS => ERROR				
A161 A0 00		LDY	#0	
A163 B1 B9		LDA	(PC), Y	
A165 B5 B1		STA	TOKCOD	
A167 C9 BE		CMP	#NUMBER	
A169 D0 1B		BNE	NCK30	
A16B A0 07		LDY	#FPREC+1	
A16D B1 B9		LDA	(PC), Y	
A16F C9 BE		CMP	#NUMBER	; NUMBER AT OTHER END?
A171 F0 03		BEG	NCK10	; YES
A173 4C 7E A3		JMP	UKERR	; CR ON DISPLAY & PRINTER, KEYERR
A176	NCK10			
A176 20 55 9F		JSR	FSTOT	
A179 20 B5 A1		JSR	PCNCHK	; SEE IF ROOM LEFT IN PRGMEM FOR #
A17C B0 06		BCS	NCK40	
A17E 20 B9 DD		JSR	FLDOR	
A181 A9 00		LDA	#0	; EQ
A183	NCK30			
A183 1B		CLC		
A184	NCK40			
A184 60		RTS		
; CHECK PC TO SEE IF ROOM IN PRGMEM FOR #				
RETURN CC => OK, CS=> NOT OK				
A185	PCNCHK			
A185 A6 B9		LDX	PC	
A187 A4 BA		LDY	PC+1	
A189 C4 D2		CPY	PC1MAX	
A18B 90 07		BCC	PCN10	
A18D E0 F9		CPX	#-FPREC-1	
A18F 90 03		BCC	PCN10	
A191	PCN05			
A191 4C 60 9C		JMP	EPERR	
A194	PCN10			
A194 E8		INX		
A195 D0 01		BNE	PCN20	
A197 C8		INX		
A198	PCN20			
A198 60		RTS		
; PC <- PC+1				
A199	PCINC			
A199 A9 01		LDA	#1	
A19B D0 02		BNE	PCADD	; JMP
A19D	PCADDN			
A19D A9 0B		LDA	#FPREC+2	; PC <- PC+FPREC+2
A19F	PCADD			
A19F 1B		CLC		
A1A0 65 B9		ADC	PC	
A1A2 90 09		BCC	PCADD1	
A1A4 A6 BA		LDX	PC+1	; INC MSB
A1A6 E8		INX		
A1A7 E4 D3		CPX	PC1MX1	; END OF MEM?
A1A9 B0 E6		BCS	PCN05	; YES. DON'T CHANGE PC
A1AB B6 BA		STX	PC+1	; NO. STORE NEW PC
A1AD	PCADD1			
A1AD B5 B9		STA	PC	; STORE LSB
A1AF 60		RTS		; RETURN CARRY CLEAR => NO ERROR

A180 PCLRO

; CLEAR FRO

A180 A9 00
A182 F0 03LDA #0
BEQ PSET0

RETURN WITH CARRY CLEAR (CC)

; JMP

A184 LDINT

; MOVE FRO TO FR1, THEN SET FRO TO A
RETURN WITH CARRY CLEAR (CC)A184 48
A185 20 B6 DD
A188 68PHA
JSR FMOVE
PLA

; FR1 <- FRO

A189 PSET0

; SET FRO TO INTEGER PASSED IN A

RETURN WITH CARRY CLEAR (CC)

A189 85 D4
A18B A9 00
A18D 85 D5
A18F 20 AA D9
A1C2 18
A1C3 60STA FRO
LDA #0
STA FRO+1
JSR IFP
CLC
RTS

; INTEGER A TO FP A

A1C4 POP0P

; POP A OFF OPSTK

A1C4 A4 9C
A1C6 D0 09
A1C8 A9 7F
A1CA 20 B7 9B
A1CD A4 9C
A1CF B0 03LDY OPPTR
BNE POP10
LDA #OSEMSG
JSR ERRSUB
LDY OPPTR
BCS POP20

; STACK UNDERFLOW

; JMP

A1D1
A1D1 88
A1D2 84 9C
A1D4
A1D4 B1 CC
A1D6 60

POP10

DEY
STY OPPTR
POP20
LDA (OPSADR),Y
RTS

A1D7 PUSH0P

; PUSH A ON OPSTK

A1D7 A4 9C
A1D9 91 CC
A1DB C8
A1DC D0 07
A1DE A9 88
A1E0 20 B7 9B
A1E3 B0 02
A1E5LDY OPPTR
STA (OPSADR),Y
INY
BNE PSH10
LDA #OSFMSG
JSR ERRSUB
BCS DSPRT2

; STACK OVERFLOW

; JMP TO RTS

A1E5 84 9C
A1E7 60

PSH10

STY OPPTR
DSPRT2 RTS

COLLEEN CALCULATOR. BY C SHAW

```

;SUBROUTINE TO DISPLAY STACK, MEM, & X
A1E8      DSOME      JSR      DSPSTK
A1E8 20 92 A7      JSR      DMEMAL
A1E8 20 CC A5
A1EE      FDSCOM      LDA      DSPFLG
A1EE A5 BD      BNE      DSPRT2      ;RETURN IF NO DISPLAY
A1F0 D0 F5      JSR      FDSPQ      ;DISPLAY FRO FOLLOWED BY "****"
A1F2 20 57 9D      ;ENTRY PT
;DAYCOM      DEC      NUMFLG      ;<- 0
A1F5 C6 A2      LDA      #COLCMD-1
A1F7 A9 15      STA      COLCRS
A1F9 85 55      LDA      #ROWCMD
A1FB A9 16      STA      ROWCRS
A1FD 85 54      LDX      #4      ;# OF CHARS
A1FF A2 04      LDA      #STARMS
A201 A9 F0      LDY      #STARMS/256
A203 A0 BA      ;PUT TEXT ON SCREEN AND PRINTER (IF OPEN)
A205      PITXTP      JSR      PTCHSP
A205 20 AB A2      ;PUT CR ON PRINTER & DISPLAY
A208      PTCRPD      JSR      PUTCRP
A208 20 9D A2
A208      PUTCR      LDA      #CR
A208 A9 9B      LDX      ROWCRS
A20D A6 54      CPX      #23
A20F E0 17      BNE      PTCHR      ;IF NOT ON BOTTOM LINE THEN DO NORMAL CR
A211 D0 1E      ;OTHERWISE, DO SCROLLING & RETURN TO BOTTOM LINE

A213 A9 10      LDA      #ROWSCR
A215 A6 BB      LDX      PROG
A217 E0 01      CPX      #STDPRG
A219 D0 02      BNE      PUTCR2
A21B A9 02      LDA      #2      ;SCROLL 22 LINES IF STORE PROGRAM MODE
A21D      PUTCR2      STA      ROWCRS
A21D 85 54      JSR      PTDEL2
A21F 20 2B A2      LDA      #23
A222 A9 17      STA      ROWCRS
A224 85 54      RTS
A226 60

A227      PUTDEL      ;DELETE BOTTOM LINE ON SCREEN
;      ;      RETURN COLCRS = LMARG = 1
A227 A9 17      LDA      #23
A229 85 54      STA      ROWCRS
A22B      PTDEL2      ;DELETE CURRENT LINE
A22B A9 01      LDA      #LMARG
A22D 85 55      STA      COLCRS
A22F A9 9C      LDA      #DELLIN
A231      PTCHR      ;PUT ONE CHAR (IN A) ON SCREEN
;
A231 A2 00      LDX      #SIOCB
A233      PTCHR2
A233 AB      TAY
A234 A5 BD      LDA      DSPFLG
A236 D0 4B      BNE      PTABC
A238 A9 0B      LDA      #PUTCHR
A23A 9D 42 03      STA      ICCOM, X
A23D A9 00      LDA      #0
A23F 9D 48 03      STA      ICBLL, X
A242 9D 49 03      STA      ICBLLH, X
A245 9B      TYA

```

```

A246 4C 55 E4      JMP      CIOV

A249      PTLINI      ;PUT UP ONE LINE OF SCREEN DISPLAY (FOR INIT)
A249 B6 9E          STX      TO
A24B BD F4 BA      LDA      CHRTAB,X
A24E 20 31 A2      JSR      PTCHR
A251 20 68 A2      JSR      CTLR16
A254 A6 9E          LDX      TO
A256 BD F5 BA      LDA      CHRTAB+1,X
A259 20 31 A2      JSR      PTCHR
A25C A2 13          LDX      #19
A25E 20 6A A2      JSR      CTLR
A261 A6 9E          LDX      TO
A263 BD F6 BA      LDA      CHRTAB+2,X
A266 D0 C9          BNE      PTCHR      ; JMP

A268      CTLR16      ;PUT 16 CTRL R'S ON SCREEN (HORIZ. LINES)
A268 A2 10          LDX      #16
A26A      CTLR          ;PUT X CTRL R'S ON SCREEN
A26A A9 3C          LDA      #CTLR5
A26C D0 04          BNE      PUTCTL      ; JMP

A26E      BLNK15      ;PUT 15 BLANKS ON SCREEN
A26E A2 0F          LDX      #15
A270      BLNKS          ;PUT X BLANKS ON SCREEN
A270 A9 28          LDA      #BLKBUF
A272      PUTCTL          LDY      #BLKBUF/256
A272 A0 05

A274      PUTCHS      ;
A274      ;           A=ICBAL, Y=ICBAH, X=# OF CHARS
A274 48          PHA
A275 8A          TXA
A276 A2 00          LDX      #SIOCB
A278 9D 48 03      STA      ICBLL,X
A27B 68          PLA
A27C      PTCHS2
A27C 9D 44 03      STA      ICBAL,X
A27F A5 BD          LDA      DSPFLG
A281 F0 03          BEQ      PTABD
A283 A0 01          PIABC    LDY      #SUCCES      ;DON'T PRINT => ALWAYS SUCCESSFUL
A285 60          RETN2    RTS
A286      PTABD
A286 98          TYA
A287 9D 45 03      STA      ICBAH,X
A28A A9 00          LDA      #0
A28C 9D 49 03      STA      ICBLL,X
A28F A9 0B          LDA      #PUTCHR
A291 9D 42 03      STA      ICCOM,X

A294 4C 56 E4      JMP      CIOV

A297      PTCRPN      ; IF PREVIOUS TOKEN WAS NUMBER THEN PUT CR ON PRINTER
A297 A5 A2          LDA      NUMFLG
A299 F0 EA          BEQ      RETN2
A29B C6 A2          DEC      NUMFLG      ; <- 0

A29D      PUTCRP      ;PUT CR ON PRINTER IF PRINTER IS ON

```


A299 F0 EA	BEG	RETN2	
A29B C6 A2	DEC	NUMFLG	; <- 0
A29D	PUTCRP		; PUT CR ON PRINTER IF PRINTER IS ON

COLLEEN CALCULATOR, BY C SHAW

A29D A6 95	LDX	PRNFLG	
A29F F0 E4	BEG	RETN2	
A2A1 A9 9B	LDA	#CR	
A2A3 A2 20	LDX	#PIOCB	
A2A5 20 33 A2	JSR	PTCHR2	
A2A8 4C CC A2	JMP	PRNCHK	; CHECK TO SEE IF PRINTER STILL THERE
A2AB	PTCHSP		; PUT CHARS ON SCREEN AND PRINTER (IF OPEN)
			A=ICBAL, Y=ICBAH, X=# OF CHARS
A2AB 8D 62 05	STA	ASAVE	
A2AE 8E 63 05	STX	XSAVE	
A2B1 8C 64 05	STY	YSAVE	
A2B4 20 74 A2	JSR	PUTCHS	
A2B7 A6 95	LDX	PRNFLG	
A2B9 F0 CA	BEG	RETN2	
A2BB A2 20	LDX	#PIOCB	
A2BD AD 63 05	LDA	XSAVE	
A2C0 9D 48 03	STA	ICBLL, X	
A2C3 AD 62 05	LDA	ASAVE	
A2C6 AC 64 05	LDY	YSAVE	
A2C9 20 7C A2	JSR	PTCHS2	
A2CC	PRNCHK		
A2CC C0 01	CPY	#SUCCE	; SUCCESSFUL PRINTING?
A2CE F0 B5	BEG	RETN2	; YES.
A2D0 C0 80	CPY	##80	; BREAK KEY ABORT?
A2D2 F0 B1	BEG	RETN2	; YES. OK - WILL BE HANDLED LATER
A2D4 4C 44 A7	JMP	OFFERR	; NO. CLOSE PRINTER & DISPLAY ERROR MSG

A2D7 SAVCHR

			MOVE CHAR FROM TOKBUF TO TOKTMP
A2D7 A0 00	LDY	#0	
A2D9 B1 82	LDA	(TOKPTR), Y	
A2DB A6 86	LDX	TOKTIN	
A2DD 95 84	STA	TOKTMP, X	
A2DF E6 86	INC	TOKTIN	
A2E1 60	RTS		

COLLEEN CALCULATOR. BY C SHAW

A2E2		UNPINT			; UNPACK KEYWORD - INITIALIZATION
A2E2 A9 0A	LDA	#KEYWRD-1			; GET REST OF WORD
A2E4 B5 8C	STA	PKPTR			
A2E6 A9 8E	LDA	#KEYWRD/256			
A2E8 B5 8D	STA	PKPTR+1			
A2EA A9 00	LDA	#0			
A2EC B5 80	STA	LFRT			
A2EE B5 8E	STA	KYLFRT			
A2F0 B5 8F	STA	KEYCNT			
A2F2 60	RTS				

A2F3		UNPNUM			; UNPACK KEYWORD - MIDDLE OF LOOP - FETCH & STORE LENGTH
A2F3 A0 00	LDY	#0			; SET UP Y REG FOR LDNIB
A2F5 B4 88	STY	KEYCHR			
A2F7 20 4C A1	JSR	LDNIB			

A2FA B5 89	STA	KEYLEN		
A2FC B5 8A	STA	KEYLN2		
A2FE 60	RTS			

A2FF		UNPNXT			; UNPACK KEYWORD - GOTO NEXT WORD (END OF LOOP)
------	--	--------	--	--	---

A2FF E6 8F	INC	KEYCNT		
A301 E6 89	INC	KEYLEN		
A303 A5 89	LDA	KEYLEN		
A305 4A	LSR	A		
A306 AA	TAX			
A307 A5 8E	LDA	KYLFRT		
A309 90 07	BCC	KEY50		
A30B 49 01	EDR	#1		
A30D F0 01	BEG	KEY40		
A30F E8	INX			

A310	KEY40			
A310 B5 8E	STA	KYLFRT		

A312	KEY50			
A312 B5 80	STA	LFRT		
A314 8A	TXA			
A315 18	CLC			
A316 65 8C	ADC	PKPTR		
A318 90 02	BCC	KEY60		
A31A E6 8D	INC	PKPTR+1		
A31C	KEY60			

A31C B5 8C	STA	PKPTR		
A31E 60	RTS			

A31F		UNPCK2		
A31F 4B	PHA			
A320 4C 2E A3	JMP	UNP10		
A323	UNPACK			

UNPACK WORD INTO TOKBUF,X FROM (PKPTR),LFRT
RETURNS LENGTH OF WORD IN Y

A323 86 82	STX	TOKPTR		
A325 A0 00	LDY	#0		
A327 98	TYA			
A328 48	PHA			
A329 20 4C A1	JSR	LDNIB		
A32C B5 8A	STA	KEYLN2		
A32E	UNP10			
A32E 20 24 A1	JSR	LDCHR		
A331 B4 8B	STY	LDNBSV		

A32E 20 24 A1
A331 84 8B
A32E 20 24 A1
A331 84 8B
STY
LDNBSV

COLLEEN CALCULATOR, BY C SHAW

A333 AA TAX
A334 68 PLA
A335 A8 TAY
A336 8A TXA
A337 91 82 STA (TOKPTR), Y
A339 C8 INY
A33A 98 TYA
A33B 48 PHA
A33C A4 8B LDY LDNBSV
A33E C6 8A DEC KEYLN2
A340 D0 EC BNE UNP10
A342 68 PLA
A343 A8 TAY
A344 85 82 STA TOKPTR ; # OF CHARS
A346 60 RTS

A347 UNPKEY ; UNPACK KEYWORD GIVEN TOKEN CODE IN TOKCOD
; OUTPUT: CHARS IN TOKBUF, Y=TOKPTR, CS IF ERROR

A347 A5 81 LDA TOKCOD
A349 C9 86 CMP #STAR
A34B 90 0F BCC UNKY10
A34D C9 8D CMP #EQUAL+1 ; OUT OF RANGE?
A34F B0 2D BCS UKERR ; YES.
A351 AA TAX ; NO. SPECIAL CHAR
A352 BD 46 BA LDA TOKCHR-STAR, X
A355 BD 00 05 STA TOKBUF
A358 A0 01 LDY #1
A35A D0 0E BNE UNKRTN ; JMP
A35C UNKY10
A35C 20 E2 A2 JSR UNPINT ; INITIALIZE
A35F UNPLP
A35F A5 8F LDA KEYCNT
A361 C5 81 CMP TOKCOD
A363 D0 0E BNE UNKY20
A365 A2 00 LDX #TOKBUF
A367 20 23 A3 JSR UNPACK
A36A UNKRTN
A36A 84 82 STY TOKPTR
A36C A9 9B LDA #CR
A36E 99 00 05 STA TOKBUF, Y
A371 18 CLC ; NO ERROR
A372 60 RTS

A373 UNKY20 ; CONTINUE WITH NEXT WORD
A373 20 F3 A2 JSR UNPNUM
A374 F0 04 BEQ UKERR ; END OF LIST => ERROR (SHOULDN'T HAPPEN)
A378 20 FF A2 JSR UNPNXT
A37B 4C 5F A3 JMP UNPLP

A37E UKERR
A37E 20 08 A2 JSR PTCRPD ; PUT CR ON SCREEN AND PRINTER
A381 4C B5 9B JMP KEYERR

A384	GETMN			; FETCH & STORE MEMNUM
A384 A9 2C		LDA	#MEMMSG	CARRY SET => ERROR, CLEAR => NO ERROR
A386 20 F0 9B		JSR	PUTMSG	; DISPLAY "ENTER MEMORY REGISTER 0-99"
A389 20 D4 A0		JSR	GETINT	
A38C D0 07		BNE	BITERR	; ERROR
A38E C9 64		CMP	#MEMLN	
A390 B0 03		BCS	BITERR	; ERROR
A392 B5 A3		STA	MEMNUM	; OK 0 -> MEMLN-1
A394 60		RTS		
A395	BITERR			
A395 A9 91		LDA	#BITMSG	
A397 4C B7 9B		JMP	ERRSUB	; DISPLAY ERROR MESSAGE (WILL RETURN WITH CS => ERROR)
A39A A9 05	LDFV	LDA	#5	; FRO<-FV
A39C D0 14		BNE	MEMLD0	
A39E A9 06	LDI	LDA	#6	; FRO<-I
A3A0 D0 10		BNE	MEMLD0	
A3A2 20 4E A9	ZIILDN	JSR	ZIPLI	; CALL ZIPLI TO COMPUTE (1+I) AND LOAD N
A3A5 20 B8 9F		JSR	FPUSH0	
A3A8 A9 07	LDN	LDA	#7	; FRO<-N
A3AA D0 06		BNE	MEMLD0	
A3AC A9 08	LDPMT	LDA	#8	; FRO<-PMT
A3AE D0 02		BNE	MEMLD0	
A3B0 A9 09	LDPV	LDA	#9	; FRO<-PV
A3B2	MEMLD0			; FRO <- MEM(A)
A3B2 20 BE A3		JSR	MEMLD2	
A3B5 4C B9 DD		JMP	FLDOR	
A3B8	MEMLD1			; ER1 <- MEM(A)
A3B8 20 BE A3		JSR	MEMLD2	
A3BB 4C 9B DD		JMP	FLD1R	
A3BE	MEMLD2			
A3BE B5 A3		STA	MEMNUM	; SET UP X & Y REGS TO LOAD OR STORE MEM(A)
A3C0	MEMLDR			; SET UP X & Y REGS TO LOAD OR STORE MEM(MEMNUM)
A3C0 A4 CF		LDY	MEMADR+1	
A3C2 A5 A3		LDA	MEMNUM	; MEMNUM <- MEMNUM*6 (FPREC=6)
A3C4 0A		ASL	A	
A3C5 65 A3		ADC	MEMNUM	; (CARRY IS CLEAR)
A3C7 90 01		BCC	MLD10	
A3C9 C8		INY		
A3CA	MLD10			
A3CA 0A		ASL	A	
A3CB 90 01		BCC	MLD20	
A3CD C8		INY		
A3CE	MLD20			
A3CE AA		TAX		
A3CF 60		RTS		
A3D0	MEMSUB			; SET UP FOR DIV, PRD, SUB, SUM, XCHM, SRCL
A3D0 20 B4 A3		JSR	GETMN	CARRY SET => ERROR, CLEAR => NO ERROR
A3D3 90 03		BCC	MS10	; GET MEMNUM
A3D5 68		PLA		
A3D6 68		PLA		; ERROR => RETURN 2 LEVELS UP
A3D7 60		RTS		
A3D8	MS10			

A3D3 90 00 BCC MS10
A3D5 68 PLA
A3D6 68 PLA
A3D7 60 RTS
A3D8 MS10

; ERROR => RETURN 2 LEVELS UP

COLLEEN CALCULATOR, BY C SHAW

A3D8 20 BB 9F	JSR	FPUSHO	; SAVE X ON STACK
A3DB 20 B6 DD	JSR	FMOVE	; FR1 <- X
A3DE 20 C0 A3	JSR	MEMLDR	; SET UP X & Y REGS
A3E1 20 B9 DD	JSR	FLDOR	; FRO <- MEM(MEMNUM)
A3E4 18	CLC		; INDICATE NO ERROR
A3E5 60	RTS		
A3E6	MEMMUL		; FRO <- FRO*MEM(A)
A3E6 85 A3	STA	MEMNUM	
A3E8 20 C0 A3	JSR	MEMLDR	
A3EB 20 98 DD	JSR	FLDIR	
A3EE 4C 97 AB	JMP	SFMUL	

A3F1	PIDVL	LDA	;LOAD X & Y REGS IN PREPARATION FOR LOADING REG 0 OR 1 WITH PI/2, 90 OR 100(1F GRAD)
A3F1 A9 24		CLC	#RADPI2
A3F3 18		ADC	RADFLQ
A3F4 65 FB		TAX	
A3F6 AA		LDY	#RADPI2/256
A3F7 A0 BA		RTS	
A3F9 60			
;TAKE COMPLEMENT OF BINARY			
A3FA	SCMP2	LDX	#3
A3FA A2 03			
A3FC	SCLP2	LDA	BINARY, X
A3FC B5 B0		EDR	##FF
A3FE 49 FF		STA	BINARY, X
A400 75 B0		DEX	
A402 CA		BPL	SCLP2
A403 10 F7		RTS	
A405 60			
;TAKE COMPLEMENT OF BINARY AND ADD 1			
A406	S2CMP	JSR	SCMP2
A406 20 FA A3		INC	BINARY+3
A407 E6 B3		BNE	STCRTN
A40B D0 0A		INC	BINARY+2
A40D E6 B2		BNE	STCRTN
A40F D0 06		INC	BINARY+1
A411 E6 B1		BNE	STCRTN
A413 D0 02		INC	BINARY+0
A415 E6 B0			
A417	STCRTN	RTS	
A417 60			
;SHIFT BINARY LEFT A PLACES			
A418	SLSHF2		RETURN WITH ORIGINAL PROCESSOR STATUS
			;ROTATING IN CARRY
A418 08		PHP	
A419 AA		TAX	
A41A F0 0D		BEG	SRTN
A41C	SLS05		
A41C 28		PLP	
A41D 08		PHP	
A41E 26 B3		ROL	BINARY+3
A420 26 B2		ROL	BINARY+2
A422 26 B1		ROL	BINARY+1
A424 26 B0		ROL	BINARY+0
A426 CA		DEX	
A427 D0 F3		BNE	SLS05
A429	SRTN		
A429 28		PLP	
A42A 60	SNUM50	RTS	

A42B

SNUMB

NUMBER PROCESSING: CONVERT ASCII IN TOKBUF TO FP IN FRO

A42B A5 B7
A42D F0 5ELDA DHOFLG
BEQ SNUM40

; DECIMAL

A42F A9 FF
A431 B5 B2
A433 20 B0 A1
A436

SNUM20

LDA #-1
STA TOKPTR
JSR PCLRO

; HEX BINARY OR OCT => CONVERT TO F.P.

A436 E6 B2
A438 A0 00
A43A B1 B2
A43C C9 9B
A43E D0 37INC TOKPTR
LDY #0
LDA (TOKPTR), Y
CMP #CR
BNE SNUM25

; CONTINUE

A440 20 BB 9F
A443 20 2E 9D
A446 20 9D 9FJSR FPUSH0
JSR FPBNCK
JSR FPOPO; SAVE FP #
; CONVERT FP TO BINARY (4 BYTES) & CHECK WHETHER IT'S WITHIN RANGE
; RESTORE FP #; IS # > BITBIN (2^(BITINT-1)-1) AND <= BITBN2 (2^BITINT)-1?
; IF SO, THEN NUMBER WAS MEANT AS NEGATIVE, E.G. \$FFFF

A449 A2 00

LDX #0

A44B
A44B B5 B0
A44D D5 A4
A44F 90 D9
A451 D0 07
A453 E8
A454 E0 04
A456 D0 F3
A458 F0 D0

BBLP1

LDA BINARY, X
CMP BITBIN, X
BCC SNUM50
BNE BB10
INX
CPX #4
BNE BBLP1
BEQ SNUM50; < BITBIN => RETURN
; > BITBIN => OK; CONTINUE
; = BITBIN => RETURN

A45A

BB10

LDX #0

A45A A2 00
A45C
A45C B5 B0
A45E D5 A8
A460 90 07
A462 D0 C6
A464 E8
A465 E0 04
A467 D0 F3

BBLP2

LDA BINARY, X
CMP BITBN2, X
BCC BB30
BNE SNUM50
INX
CPX #4
BNE BBLP2; < BITBN2 => OK
; > BITBN2 => RETURN

; = BITBN2 => OK

A469

BB30

LDX #3

; OK => INPUT WAS REALLY MEANT AS NEG. #

A469 A2 03
A46B
A46B B5 B0
A46D 15 AC
A46F 95 B0
A471 CA
A472 10 F7

BBLP3

LDA BINARY, X
ORA BINMIN, X
STA BINARY, X
DEX
BPL BBLP3

; OR WITH BINMIN= -(2^(BITNIT-1)) TO EXTEND SIGN BIT

A=MSB WHICH SHOULD BE NEG.
; CONVERT TO NEW FLOATING # (SHOULD BE NEG.) AND RETURN

A474 4C 27 AB

JMP BINFP2

A477

SNUM25

A477 4B
A478 A5 B7
A47A 20 B4 AB
A47D 6B
A47E 3B
A47F E9 30PHA
LDA DHOFLG
JSR INTMUL
PLA
SEC
SBC #0

; '0-'9 -> 0-9

A481 C9 11		CMP	#'A-'O	
A483 90 02		BCC	SNUM30	
A485 E9 07		SBC	#'A-'O-10	I A-F -> 10-15
A487	SNUM30			
A487 20 53 A9		JSR	INTADD	
A48A 4C 36 A4		JMP	SNUM20	
A48D	SNUM40			
A48D 20 22 9C		JSR	TKINT2	I A <- #TOKBUF, Y <- #TOKBUF/256
A490 85 F3		STA	INBUFF	
A492 84 F4		STY	INBUFF+1	
A494 A9 00		LDA	#0	
A496 85 F2		STA	CIX	
A498 4C 00 DB		JMP	AFP	

ROUTINES CORRESPONDING TO KEYWORDS

```

;
A49B      RETURN
A49B 60    RTS
A49C      SAC30
A49C A5 D4 LDA    FRO
A49E D0 06 BNE    SAC30
;
A4A0      SAC10
A4A0 20 F1 A3 JSR    PIOVL
A4A3 4C 89 DD JMP    FLDOR
;
A4A6      SAC30
A4A6 48    PHA
A4A7 20 8C 9F JSR    ARCSUB
A4AA A5 E0 LDA    FR1
A4AC D0 1D BNE    SAC34
A4AE 68    PLA
A4AF 30 03 BMI    S180PI
A4B1 4C B0 A1 JMP    PCLRO
A4B4      S180PI
A4B4 A6 FB LDX    RADFLG
A4B6 F0 0C BEQ    SPI10
A4B8 A9 B4 LDA    #180
;
; CPX    #GRADON
; BNE    SAC32
; LDA    #200
; SAC32
A4BA 4C B9 A1 JMP    PSETO
A4BD      SPI
A4BD A5 94 LDA    RPNALG
A4BF D0 03 BNE    SPI10
A4C1 20 BB 9F JSR    FPUSHO
A4C4      SPI10
A4C4 A2 00 LDX    #PICONST
A4C6 A0 BA LDY    #PICONST/256
A4C8 4C B9 DD JMP    FLDOR
;
A4CB      SAC34
A4CB 20 3A A9 JSR    SFDIV
A4CE 20 24 A9 JSR    SRECIP
A4D1 20 1B B1 JSR    SATAN
A4D4 68    PLA
A4D5 10 C4 BPL    RETURN
A4D7 20 B6 DD JSR    FMOVE
A4DA 20 B4 A4 JSR    S180PI
A4DD 4C 6A A9 JMP    SFADD
; ARCCOS(FRO) = ARCTAN(SQRT(1-FRO*FRO)/FRO)
; ARCCOS(0) = 90 DEG = PI/2 RAD. SPECIAL CASE BECAUSE TAN UNDEFINED
; LOAD X & Y REGS TO GET PI/2, 90 OR 100
; FR1 <- SQRT(1-FRO*FRO)
; ABSVAL(FRO) = 1
; FRO=+1. ARCCOS(+1) = 0
; FRO <- 180 OR 200 OR PI, DEPENDING ON RADFLG
; FRO<-PI
; DEG => 180
; GRAD => 200
; X <- PI
; PUSH OLD X IF RPN
; PICONST
; PICONST/256
; COS <0 => ADD 180 DEG OR 200 GRAD OR PI TO ARCCOS

```

A4E0	SBITS			
				; SET OCTAL, HEX WORD LENGTH TO 1-32 BITS
				BINARY TO 1-16 BITS
				; DISPLAY "ENTER 1-32"
A4E0 A9 20		LDA	#BTMSG0	
A4E2 20 F0 9B		JSR	PUTMSG	
A4E3 20 D4 A0		JSR	GETINT	; GET INTEGER
A4E8 B0 07		BCS	SBERR	; NOT GOOD - ERROR ALREADY REPORTED
A4EA AA		TAX		
A4EB F0 04		BEQ	SBERR	; TOO SMALL
A4ED C9 21		CMP	#32+1	
A4EF 90 03		BCC	SBITS2	; TOO LARGE?
A4F1	SBERR			
A4F1 4C 95 A3		JMP	BITERR	
A4F4	SBITS2			
A4F4 B5 9D		STA	BITINT	
A4F6 AA		TAX		
A4F7 CA		DEX		
A4F8 8A		TXA		
A4F9 A2 00		LDX	#0	
A4FB B6 B3		STX	BINARY+3	
A4FD B6 B2		STX	BINARY+2	
A4FF B6 B1		STX	BINARY+1	
A501 B6 B0		STX	BINARY+0	; SET BINARY TO 0
A503 38		SEC		
A504 20 18 A4		JSR	SLSHF2	; SHIFT LEFT BITINT BITS WITH CARRY
A507 A2 03		LDX	#3	
A509	SBLP1			
A509 B5 B0		LDA	BINARY, X	
A50B 95 A4		STA	BITBIN, X	
A50D CA		DEX		
A50E 10 F9		BPL	SBLP1	
A510 20 FA A3		JSR	SCMP2	; TAKE COMP
A513 A2 03		LDX	#3	
A515	SBLP2			
A515 B5 B0		LDA	BINARY, X	
A517 95 AC		STA	BINMIN, X	
A519 CA		DEX		
A51A 10 F9		BPL	SBLP2	
A51C 20 FA A3		JSR	SCMP2	
A51F A9 01		LDA	#1	
A521 38		SEC		
A522 20 18 A4		JSR	SLSHF2	; (2^BITINT)-1
A525 A2 03		LDX	#3	
A527	SBLP3			
A527 B5 B0		LDA	BINARY, X	
A529 95 AB		STA	BITBN2, X	
A52B CA		DEX		
A52C 10 F9		BPL	SBLP3	
A52E A2 B0		LDX	#'0+'\$B0	; CONVERT INT 0-99 TO CHAR 00-99 (INVERSE VIDEO)
A530 8E 00 05		STX	TOKBUF	
A533 A5 9D		LDA	BITINT	
A535	DSTLP			
A535 C9 0A		CMP	#10	
A537 90 07		BCC	DST10	
A539 E9 0A		SBC	#10	; (CARRY SET)
A53B EE 00 05		INC	TOKBUF	
A53E D0 F5		BNE	DSTLP	; JMP
A540	DST10			
A540 69 B0		ADC	#'0+'\$B0	; (CARRY CLEAR) (INVERSE VIDEO)

A540 69 00

ADC

#'0+\$80

; (CARRY CLEAR) (INVERSE VIDEO)

COLLEEN CALCULATOR, BY C SHAW

```

A542 8D 01 05      STA      TOKBUF+1      ; LSDIGIT
A545 A2 02          LDY      #2             ; 2 CHARS
A547 A9 13          LDA      #DBITS

A549                DSPST
A549 85 55          STA      COLCRS
A54B                DSPST2
A54B A9 01          LDA      #ROWSTT
A54D 85 54          STA      ROWCRS
A54F 20 78 A5       JSR      DSPCLR        ; CLEAR DSPFLG Y<-0  A<-OLD DSPFLG
A552 48             PHA                     ; & SAVE OLD DSPFLG => ALWAYS DISPLAY
A553 98             TYA                     ; 0 = LSB OF ADDR
A554 A0 05          LDY      #TOKBUF/256
A556 20 74 A2       JSR      PUTCHS
A559                DSPLOD
A559 68             PLA                     ; LOAD OLD DSPFLG & RESTORE
A55A 85 BD          STA      DSPFLG        ; RESTORE OLD DSPFLG
A55C 60             RTS

```

```

A55D                SFIX
A55D A9 16          LDA      #FIXMSG        ; DISPLAY "ENTER 0-8"
A55F 20 F0 9B       JSR      PUTMSG
A562 20 D4 A0       JSR      GETINT        ; GET NEXT TOKEN: WANT INTEGER 0-8
A565 B0 8A          BCS      SBERR        ; ERROR
A567 C9 09          CMP      #8+1
A569 B0 86          BCS      SBERR        ; ERROR - TOO LARGE
A56B                SFIX2
A56B 85 C0          STA      FIXNUM
A56D 69 B0          ADC      #'0+$80      ; -> ASCII (INVERSE VIDEO)
A56F BD 00 05       STA      TOKBUF
A572 A9 19          LDA      #DFIX
A574 A2 01          LDY      #1
A576 D0 D1          BNE      DSPST        ; DISPLAY IN STATUS AREA OF SCREEN
                                           ; # OF CHARS
                                           ; JUMP TO DISPLAY STATUS

```

```

A57B                DSPCLR
A57B A5 BD          LDA      DSPFLG        ; CLEAR DSPFLG & LOAD OLD DSPFLG INTO A
A57A A0 00          LDY      #0           ; A <- OLD DSPFLG  X UNCHANGED  Y <- 0
A57C 84 BD          STY      DSPFLG
A57E 60             RTS

```

```

A57F                SCOMPL
A57F 20 51 A9       JSR      ONEADD        ; X <- COMPLEMENT(X) = -(X+1)
                                           ; X<-X+1

```

```

A582                SCHGSG
A582 A5 D4          LDA      FRO          ; FRO<- -FRO
A584 F0 04          BEQ      SCH10
A586 49 80          EOR      #$80
A588 85 D4          STA      FRO
A58A                SCH10
A58A 60             RTS

```

```

A58B                SCLINI

```

A58B A2 21	LDX	#ENTER	; GOTO ENTER MODE AND CHANGE STATUS LINE
A58D B6 B1	STX	TOKCOD	
A58F 20 0B AE	JSR	SENER	
A592 A9 9F	LDA	#INTCHR-3	; CLEAR MEM4-9 FOR INTEREST CALCS & DISPLAY TITLES
A594 A0 1B	LDY	#FPREC*4	
A596 D0 04	BNE	CLST10	; JMP
A598	SCLSTA		; CLEAR MEM 3-9 FOR STATISTICS & DISPLAY TITLES
A598 A9 B4	LDA	#STACHR-3	
A59A A0 12	LDY	#FPREC*3	
A59C	CLST10		
A59C B5 9F	STA	T1	
A59E A9 00	LDA	#0	; CLEAR MEM
A5A0	CLSLP1		
A5A0 91 CE	STA	(MEMADR), Y	
A5A2 C8	INY		
A5A3 C0 3C	CPY	#FPREC*10	
A5A5 90 F9	BCC	CLSLP1	
A5A7 A9 0B	LDA	#ROWREG+3	; START WITH MEM 3
A5A9 B5 54	STA	ROWCRS	
A5AB A9 06	LDA	#6	
A5AD B5 9E	STA	T0	
A5AF	CLSTLP		
A5AF A9 14	LDA	#20	; DISPLAY TITLES IN COL 20
A5B1 B5 55	STA	COLCRS	
A5B3 A5 9F	LDA	T1	
A5B5 1B	CLC		
A5B6 69 03	ADC	#3	
A5B8 B5 9F	STA	T1	
A5BA A0 BA	LDY	#INTCHR/256	
A5BC A2 03	LDX	#3	
A5BE 20 74 A2	JSR	PUTCHS	; DISPLAY CHARS ON SCREEN ONLY
A5C1 E6 54	INC	ROWCRS	; GOTO NEXT LINE ON SCREEN
A5C3 C6 9E	DEC	T0	
A5C5 10 EB	BPL	CLSTLP	; CONTINUE
A5C7 30 03	BMI	DMEMAL	; DONE - JMP TO DISPLAY NEWLY ZEROED MEM
A5C9	SCLMEM		; CLEAR MEMORY
A5C9 20 03 AA	JSR	MEMCLR	
A5CC	DMEMAL		; DISPLAY ALL OF MEM (0-9)
A5CC A5 BD	LDA	DSPFLG	
A5CE D0 BA	BNE	SCH10	; RETURN IF NO DISPLAY
A5D0 20 BB 9F	JSR	FPUSHO	; SAVE X
A5D3 A9 00	LDA	#0	
A5D5 20 B2 A3	JSR	MEMLDO	; FRO<-MEM(MEMNUM) (USING MEMNUM)
A5D8 20 B6 9D	JSR	TOKNUM	; TOKBUF<-ASCII(FRO)
A5DB 20 72 AB	JSR	DSPMEM	; DISPLAY IN MEM AREA OF SCREEN
A5DE A6 A3	LDX	MEMNUM	
A5E0 E8	INX		
A5E1 BA	TXA		
A5E2 C9 0A	CMR	#10	
A5E4 90 EF	BCC	DMELP	; CONTINUE
A5E6 4C 9D 9F	JMP	FPOPO	; DONE - RELOAD X

COLLEEN CALCULATOR, BY C SHAW

A1B0	SCLX	=	PCLRO	
A5E9	SFACTO			; X! = X(X-1)(X-2).....
A5E9 20 BB 9F	JSR	FPUSHO		; GINT2 WILL POP
A5EC 20 F2 A0	JSR	GINT2		; A= INTEGER 0-255
A5EF B0 21	BCS	SFERR		; ERROR
A5F1 C9 45	CMP	#69		
A5F3 B0 1D	BCS	SFERR		; ERROR - TOO LARGE
A5F5 AA	TAX			
A5F6 D0 02	BNE	SF10		
A5F8 A9 01	LDA	#1		; 0! = 1! = 1
A5FA	SF10			
A5FA 48	PHA			
A5FB 20 B9 A1	JSR	PSETO		; FACT <- N
A5FE	SFLP			
A5FE 20 B6 DD	JSR	FMOVE		
A601 68	PLA			
A602 C9 03	CMP	#3		
A604 90 0F	BCC	SFDDN		
A606 E9 01	SBC	#1		; X <- X-1
A608 48	PHA			
A609 20 B9 A1	JSR	PSETO		; INT -> FP
A60C 20 DB DA	JSR	FMUL		; FACT <- FACT * X
A60F 90 ED	BCC	SFLP		
A611 68	PLA			; CARRY SET => MULTIPLY ERROR (SHOULDN'T HAPPEN)
A612	SFERR			
A612 4C B6 A9	JMP	CRYCHK		; FRO <- 0 "ARITHMETIC OVERFLOW"
A615	SFDDN			
A615	SPOW60			
A615 60	RTS			
A616	SROOT			
A616 20 24 A9	JSR	SRECIP		; Y ROOT X = Y POWER 1/X
A619	SPOWER			; Y^X = EXP10(X* LOG10(Y))
A619 20 86 9F	JSR	FPOP1		
A61C A5 E0	LDA	FR1		
A61E 30 06	BMI	SPOW25		; Y < 0 => ERROR STOP
A620 D0 0A	BNE	SPOW40		
A622 A5 D4	LDA	FRO		; Y=0
A624 10 03	BPL	SPOW30		
A626 4C 88 A9	JMP	CRYSDN		; X<0 => ERROR CLEAR FRO
A629	SPOW25			
A629 4C B0 A1	JMP	PCLRO		
A62C	SPOW40			
A62C 20 5C 9F	JSR	FST1T		
A62F A5 D4	LDA	FRO		; SAVE SIGN OF X
A631 48	PHA			
A632 29 7F	AND	#\$7F		
A634 85 D4	STA	FRO		; X <- !X!
A636 20 BB 9F	JSR	FPUSHO		
A639 A9 01	LDA	#1		
A63B 85 A1	STA	INTFLG		
A63D 20 7A 49	JSR	SFRACT		; TAKE FRACTIONAL PART IF =0 THEN X IS AN INTEGER

COLLEEN CALCULATOR, BY C SHAW

A640 A5 D4		LDA	FRO	
A642 D0 0C		BNE	SPOW50	
A644 20 49 9F		JSR	FLDOT	; X IS INTEGER
A647 20 7A A9		JSR	SFRACT	; TAKE FRACTIONAL PART
A64A A5 D4		LDA	FRO	
A64C D0 02		BNE	SPOW50	
A64E C6 A1		DEC	INTFLG	; Y IS INTEGER
A650	SPOW50			
A650 20 49 9F		JSR	FLDOT	; Y
A653 20 BE A6		JSR	SLOGTE	
A656 20 94 A8		JSR	SPMUL	; X * LOGTEN(Y)
A659 20 07 98		JSR	SEXPTE	
A65C A5 A1		LDA	INTFLG	; BOTH X & Y INTEGER?
A65E D0 17		BNE	SPOW80	; NO
A660	SROUND			; ROUND(X) = SIGN(X)*TRUNC(ABS(X)+.5)
A660 A2 6C		LDX	#FHALF	
A662 A0 DF		LDY	#FHALF/256	
A664 20 98 DD		JSR	FLD1R	
A667 A5 D4		LDA	FRO	
A669 10 06		BPL	SROU10	
A66B 20 83 A9		JSR	SFSUB	; NEG => SUBTRACT .5
A66E 4C 74 A6		JMP	SPOW70	
A671	SROU10			
A671 20 6A A9		JSR	SFADD	
A674	SPOW70			
A674 20 7D A6		JSR	STRUNC	; TRUNCATE FRO
A677	SPOW80			
A677 68		PLA		; LOAD SIGN OF X
A67B 10 9B		BPL	SPOW60	; POSITIVE => RETURN
A67A 4C 24 A9		JMP	SRECIP	; NEGATIVE => 1/X
A67D	STRUNC			
A67D 20 95 A6		JSR	XINT	; PERFORM INT FUNCTION (ALMOST)
A680	XINT4			
A680 4C 00 DC		JMP	NORM	; NORMALIZE (TRUNCATE)
A683	SINTEG			
A683 20 95 A6		JSR	XINT	; FRO <- INT(FRO)
A686 A6 D4		LDX	FRO	; INTEGER SUBROUTINE
A68B 10 F6		BPL	XINT4	
A68A AA		TAX		
A68B F0 F3		BEQ	XINT4	
A68D	SUBONE			
A68D A9 01		LDA	#1	; FRO <- FRO-1
A68F	SUBINT			
A68F 20 75 A9		JSR	INTSUB	; FRO <- FRO - A
A692 4C 82 A5		JMP	SCHGSG	
A695	; XINT			INT ROUTINE FROM SHEP ATARI BASIC B0D5-B0EE
A695 A5 D4		LDA	FRO	; GET EXPONENT
A697 29 7F		AND	#\$7F	; AND OUT SIGN BIT

A697 29 7F

AND

#\$7F

; AND OUT SIGN BIT

COLLEEN CALCULATOR, BY C SHAW

A699 38		SEC		
A69A E9 3F		SBC	#\$3F	; GET LOCATION OF 1ST FRACTION BYTE
A69C 10 02		BPL	XINT1	; IF >= 0 THEN BRANCH
A69E A9 00		LDA	#0	; ELSE SET =0
A6A0	XINT1			
A6A0 AA		TAX		; PUT IN X AS INDEX INTO FROM
A6A1 A9 00		LDA	#0	; SET ACCUM TO ZERO FOR ORING
A6A3 AB		TAY		; ZERO Y
A6A4	INT2			
A6A4 E0 05		CPX	#FPREC-1	; IS D.P. LOC >= 5?
A6A6 B0 07		BCS	INTRTN	; IF YES, LOOP DONE
A6A8 15 D5		ORA	FRO+1, X	; OR IN THE BYTE OF MANTISSA
A6AA 94 D5		STY	FRO+1, X	; ZERO BYTE
A6AC EB		INX		; POINT TO NEXT BYTE
A6AD D0 F5		BNE	INT2	; JMP
A6AF	INTRTN			
A6AF 60		RTS		

A6B0	ZLN11			; LN(1+I)
A6B0 20 4E A9		JSR	Z1PLI	

A6B3	SLN			; FRO <- LN(FRO)
A6B3 20 C9 A6		JSR	LOGCHK	; CHECK FOR 0,1 (SPECIAL CASES)
A6B6 B0 03		BCS	GOCRY	
A6B8 20 CD DE		JSR	LOG	
A6BB	GOCRY			
A6BB 4C B6 A9		JMP	CRYCHK	

A6BE	SLOGTE			; FRO <- LOG10(FRO)
A6BE 20 C9 A6		JSR	LOGCHK	
A6C1 B0 F8		BCS	GOCRY	
A6C3 20 D1 DE		JSR	LOG10	
A6C6 4C B6 A9		JMP	CRYCHK	

A6C9	LOGCHK			; CHECK FOR 0,1
A6C9 38		SEC		
A6CA A5 D4		LDA	FRO	
A6CC F0 E1		BEG	INTRTN	; LN(0), LOG(0) => ERROR
A6CE A2 05		LDX	#FPREC-1	
A6D0	LOGCLP			
A6D0 B5 D4		LDA	FRO, X	
A6D2 DD 42 BA		CMP	ONE, X	
A6D5 18		CLC		
A6D6 D0 D7		BNE	INTRTN	; NOT 1 => OK
A6D8 CA		DEX		
A6D9 10 F5		BPL	LOGCLP	
A6DB 68		PLA		; SKIP LOGCHK RETURN
A6DC 68		PLA		
A6DD 4C B0 A1		JMP	PCLRO	; LN(1)=LOGTEN(1)=0

A6E0	INTMOD			; FRO <- FRO MOD A (ALSO MODFAC <- INT(Y/X))
------	--------	--	--	--

A6E0 48		PHA	
A6E1 20 BB 9F		JSR	FPUSHO
A6E4 68		PLA	
A6E5 20 B9 A1		JSR	PSETO

A6E8		SHOD			; Y MOD X = Y-X*INT(Y/X)
A6E9	20 B6 DD	JSR	FMOVE		
A6EB	20 18 9F	JSR	FLDOS		
A6EE	20 EB 9F	JSR	FPUSH1		
A6F1	20 3A A9	JSR	SF DIV		; Y/X
A6F4	20 B3 A6	JSR	SINTEG		; INT(Y/X)
A6F7	A2 50	LDX	#MODFAC		
A6F9	A0 05	LDY	#MODFAC/256		
A6FB	20 A7 DD	JSR	FSTOR		; SAVE INT(Y/X) IN MODFAC
A6FE	20 94 AB	JSR	SPMUL		; INT(Y/X)*X
A701	4C 80 A9	JMP	SPSUB		; Y - INT(Y/X)*X

COLLEEN CALCULATOR. BY C SHAW

A704 A9 00	SDEC	LDA	#0	; DECIMAL MODE
A706 F0 06		BEG	SOCT10	; JMP
A708 A9 10	SHEX	LDA	#16	; HEXADECIMAL (BASE 16)
A70A D0 02		BNE	SOCT10	; JMP
A70C	SOCT			; SET OCTAL MODE
A70E A9 08		LDA	#8	
A70E	SOCT10			
A70E 85 87		STA	DHDFLG	
A710 A9 0B		LDA	#DDEC	
A712 20 5D A7		JSR	CHSTAT	; CHANGE STATUS LINE ON SCREEN
A715 4C CC A5		JMP	DMEMAL	; DISPLAY MEMORY IN NEW BASE
A718	SADV			
A718 A6 95		LDX	PRNFLG	; PUT CR ON PRINTER
A71A F0 0A		BEG	SADV20	; PRINTER ON ALREADY?
A71C	SADV10			OUTPUT CR & RETURN
A71C 20 78 A5		JSR	DSPCLR	
A71F 48		PHA		; CLEAR DSPFLG
A720 20 9D A2		JSR	PUTCRP	
A723 4C 59 A5		JMP	DSPLOD	
A726	SADV20			; RESTORE DSPFLG
A726 20 37 A7		JSR	SON	
A729 B0 27		BCS	POPRTN	; NO. TURN ON
A72B 20 1C A7		JSR	SADV10	; ERROR => RETURN
A72E	SOFF			; OUTPUT CR
A72E A2 00		LDX	#0	; CLOSE PRINTER
A730 86 95		STX	PRNFLG	
A732 A2 20		LDX	#PIOCB	; ALWAYS OFF
A734 4C F2 AC		JMP	XCLOSE	
A737	SON			; CLOSE X AND CALL CIO
A737 A6 95		LDX	PRNFLG	; OPEN PRINTER FOR OUTPUT
A739 D0 16		BNE	POPRTN	
A73B A2 20		LDX	#PIOCB	; ALREADY OPEN
A73D A0 04		LDY	#4	
A73F 20 F6 AC		JSR	CIOINT	
A742 F0 09		BEG	POPRTN	; SET UP IOCB AND CALL CIO & CHECK FOR SUCCESS
A744	OFFERR			; SUCCESSFUL
A744 98		TYA		; NOT SUCCESSFUL
A745 48		PHA		
A746 20 2E A7		JSR	SOFF	; SAVE ERROR #
A749 68		PLA		
A74A 4C AA AC		JMP	IDERR2	; RELOAD ERROR #
A74D	POPRTN			; DISPLAY "ERROR -" I/O ERROR #
A74D A2 01		LDX	#1	
A74F 86 95		STX	PRNFLG	
A751	POPRTN			
A751 18		CLC		; NO ERROR
A752 60	POPRTN	RTS		
9F9D	SPOP	=	FPOPO	
9FBB	SPUSH	=	FPUSHO	; POP # OFF STACK
A753 A9 06	SDEC	LDA	#6	; PUSH # ON STACK
A755 D0 02		BNE	SRAD10	
A757	SRAD			
A757 A9 00		LDA	#0	; SET RAD MODE
A759	SRAD10			; RADON
A759 85 FB		STA	RADFLG	
A75B A9 07		LDA	#DDEG	
				; CHANGE STATUS LINE ON SCREEN

```

A75D          CHSTAT          STA    COLCRS      ; CHANGE STATUS BY DISPLAYING KEYWORD AT A ON STATUS LINE
A75D 85 55
A75F 20 47 A3      JSR    UNPKY
A762 20 27 9C      JSR    INVID      ; INVERSE VIDED
A765 A6 B2        LDX    TOKPTR
A767 E0 05        CPX    #5
A769 B0 06        BCS    CHS30      ; IF >= 5 CHARS THEN NO BLANK
A76B A9 20        LDA    #'        ; ADD ONE BLANK TO CLEAR LONGER WORDS
A76D 9D 00 05      STA    TOKBUF, X
A770 EB          INX
A771          CHS30
A771 4C 4B A5      JMP    DSPST2     ; DISPLAY TOKBUF (X CHARS) ON ROWSTT LINE

```

```

A774          SCLR          JSR    PCLRO      ; CLEAR X, STACK
A774 20 B0 A1      BCC    SCLSTK      ; JMP
A777 90 11
A779          SALG          LDA    #ALGP
A779 A9 01        BNE    SRPN10      ; JMP
A77B D0 06
A77D          SALGN        LDA    #ALGNOP
A77D A9 02        BNE    SRPN10      ; JMP
A77F D0 02
A781          SRPN         LDA    #0      ; RPN
A781 A9 00
A783          SRPN10
A783 85 94        STA    RPNALG
A785 A9 02        LDA    #DALG
A787 20 5D A7      JSR    CHSTAT      ; CHANGE STATUS LINE ON SCREEN
A78A          SCLSTK
A78A A9 01        LDA    #1      ; CLEAR STACKS (LPAD ONLY)
A78C 85 9C        STA    DPPTR
A78E A9 00        LDA    #0      ; X ONLY, NOTHING ON STACK
A790 85 9B        STA    FPPTR
A792          DSPSTK
A792 A5 BD        LDA    DSPFLG
A794 D0 BC        BNE    POPRTN      ; DON'T DISPLAY - RETURN
A796 20 BB 9F      JSR    FPUSHO      ; DISPLAY STACK   SAVE X ON STACK
A799 A9 05        LDA    #ROWREG
A79B 85 54        STA    ROWCRS
A79D A5 9B        LDA    FPPTR
A79F 38          SEC
A7A0 E9 06        SBC    #FPREC
A7A2          STKD10
A7A2 85 9E        STA    TO
A7A4 AA          TAX
A7A5 A0 06        LDY    #FPSTK/256
A7A7 20 89 DD      JSR    FLDOR
A7AA A9 04        LDA    #4      ; DISPLAY IN COLUMN 4
A7AC 20 7B 9D      JSR    FDSP1
A7AF E6 54        INC    ROWCRS
A7B1 A5 54        LDA    ROWCRS
A7B3 C9 0F        CMP    #ROWSCR-1
A7B5 B0 1D        BCS    STKD45      ; STACK AT LEAST 10 DEEP
A7B7 A5 9E        LDA    TO
A7B9 38          SEC

```


COLLEEN CALCULATOR, BY C SHAW

A7BA E9 06		SBC	#FPREC	
A7BC B0 E4		BCS	STKD10	; CONTINUE
A7BE A5 54		LDA	ROWCRS	
A7C0 4B		PHA		; SAVE NEW PRVSTK
A7C1	STKD30			
A7C1 A5 54		LDA	ROWCRS	
A7C3 CD 65 05		CMP	PRVSTK	; CLEAR ALL ROWS UP TO PREVIOUS STACK MAX
A7C6 B0 0B		BCS	STKD40	
A7C8 A9 03		LDA	#3	
A7CA B5 55		STA	COLCRS	
A7CC 20 6E A2		JSR	BLNK15	
A7CF E6 54		INC	ROWCRS	
A7D1 D0 EE		BNE	STKD30	; JMP
A7D3	STKD40			
A7D3 6B		PLA		
A7D4	STKD45			
A7D4 8D 65 05		STA	PRVSTK	
A7D7 4C 9D 9F		JMP	FPOPO	
A7DA A9 00	SLSHF	LDA	#0	; Y LSHF X
A7DC F0 02		BEG	SHFSUB	; JMP
A7DE	SRSHF			; Y RSHF X
A7DE A9 01		LDA	#1	
A7E0	SHFSUB			
A7E0 B5 9E		STA	TO	; Y SHF X (RIGHT OR LEFT)
A7E2 A5 D4		LDA	FRO	; 1=>RIGHT, 0=> LEFT
A7E4 10 0A		BPL	SHF05	
A7E6 29 7F		AND	#\$7F	
A7E8 B5 D4		STA	FRO	; X<0: TAKE ABSOLUTE VALUE AND SHIFT IN OPPOSITE DIRECTION
A7EA A5 9E		LDA	TO	
A7EC 49 01		EOR	#1	
A7EE B5 9E		STA	TO	
A7F0	SHF05			
A7F0 20 D2 D9		JSR	FPI	; FP -> INT
A7F3 0B		PHP		
A7F4 20 9A 9F		JSR	FMVPOP	; FR1 <- FRO (X), POP Y
A7F7 2B		PLP		; RELOAD CARRY FROM FPI
A7FB B0 0A		BCS	SHF10	; ERROR => RETURN 0 (VERY LARGE SHIFT)
A7FA A5 E1		LDA	FR1+1	
A7FC D0 06		BNE	SHF10	; SHIFT > 256 => RETURN 0 (LARGE SHIFT)
A7FE A5 E0		LDA	FR1	; LOAD LSB OF SHIFT
A800 C5 9D		CMP	BITINT	; SHIFT > MAX # OF BITS ALLOWED IN #?
A802 90 03		BCC	SHF15	; NO. CONTINUE
A804	SHF10			
A804 4C B0 A1		JMP	PCLRO	; YES. SAVE TIME BY NOT DOING SHIFT - CLEAR & RETURN
A807	SHF15			
A807 4B		PHA		; SAVE
A80B 20 E6 9C		JSR	FPBIN	
A80B 6B		PLA		
A80C AA		TAX		
A80D A4 9E		LDY	TO	
A80F D0 06		BNE	SHF20	; RIGHT
A811 1B		CLC		; LEFT
A812 20 1B A4		JSR	SLSHF2	; SHIFT LEFT A BITS WITH CARRY: RETURN ORIGINAL PROCESSOR STATUS
A815 90 0B		BCC	BINFP	; JMP
A817	SHF20			
A817 46 B0		LSR	BINARY+0	; RIGHT
A819 66 B1		ROR	BINARY+1	
A81B 66 B2		ROR	BINARY+2	

A81D 66 B3		ROR	BINARY+3	
A81F CA		DEX		
A820 D0 F5		BNE	SHF20	
A822	BINFP			, 4 BYTE BINARY TO FP
A822 20 FC 9C		JSR	BINCHK	, CHECK IF BINARY <= BITINT BITS. IF NOT THEN BINARY <- 0
A825 A5 B0		LDA	BINARY	
A827	BINFP2			
A827 85 A0		STA	NEGFLG	
A829 10 03		BPL	BIN10	
A82B 20 06 A4		JSR	S2CMP	, TAKE ABSOLUTE VALUE
A82E	BIN10			
A82E A5 B0		LDA	BINARY	, 65536*IFP(BINARY+1, BINARY)+IFP(BINARY+3, BINARY+2)
A830 85 D5		STA	FRO+1	
A832 A5 B1		LDA	BINARY+1	
A834 85 D4		STA	FRO	
A836 20 AA D9		JSR	IFP	
A839 A2 30		LDX	#C65536	
A83B 20 C3 AD		JSR	LDY1ML	, FRO <- FRO*65536
A83E 20 88 9F		JSR	FPUSH0	
A841 A5 B2		LDA	BINARY+2	
A843 85 D5		STA	FRO+1	
A845 A5 B3		LDA	BINARY+3	
A847 85 D4		STA	FRO	
A849 20 AA D9		JSR	IFP	
A84C 20 67 A9		JSR	SPADD	
A84F A5 A0		LDA	NEGFLG	
A851 10 06		BPL	BIN20	
A853 A5 D4		LDA	FRO	, NEGATIVE #
A855 09 80		DRA	#80	
A857 85 D4		STA	FRO	
A859	BIN20			
A859	SHF30			
A859 60		RTS		
A85A	SSQUAR			
A85A 20 B6 DD		JSR	FMOVE	, X SQUARED = X*X
A85D 4C 97 AB		JMP	SFMUL	
A860	SST0			
A860 20 84 A3		JSR	GETMN	, MEM <- X
A863 B0 F4		BCS	SHF30	, ERROR => RETURN
A865	SST010			
A865 20 C0 A3		JSR	MEMLDR	
A868 20 A7 DD		JSR	FSTOR	
A86B A5 BD		LDA	DSPFLG	, DISPLAY INHIBIT?
A86D D0 EA		BNE	SHF30	, YES. NO DISPLAY AT ALL
A86F 20 EE A1		JSR	FDSCOM	, YES. NO DISPLAY, JUST CONVERT FP TO ASCII
A872	DSPMEM			, DISPLAY MEMNUM ON SCREEN (ASCII ALREADY IN TOKBUF)
A872 A5 A3		LDA	MEMNUM	
A874 20 9D AB		JSR	DSPM3	
A877 B0 2D		BCS	DM10	, MEM >= 10 SO DON'T DISPLAY
A879 A5 BD		LDA	DSPFLG	
A87B D0 29		BNE	DM10	, RETURN IF NO DISPLAY
A87D A9 18		LDA	#24	, COLUMN # FOR MEM DISPLAY
A87F B5 55		STA	COLCRS	
A881 4C B0 9D		JMP	FDSP2	, MEM < 10 SO DISPLAY
A884	INTMUL			
A884 20 B4 A1		JSR	LDINT	, FRO <- A*FRO
A887 90 0E		BCC	SFMUL	, FR1 <- FRO, FRO<-A
A889	ZMUL1I			, JMP
A889 A5 C5		LDA	DUEFLG	, IF ANNUITY DUE THEN FRO<-FRO*(1+I)

A889 ZMUL11 IF ANNUITY DUE THEN FRO<-FRO*(1+I)
A889 A5 C5 LDA DUEFLG

COLLEEN CALCULATOR, BY C SHAW

```

A88B 6A          ROR    A
A88C B0 1B      BCS    ZMRTN
A88E 20 B8 9F   JSR    FPUSHO
A891 20 4E A9   JSR    Z1PLI
A894          SPMUL
A894 20 9A 9F   JSR    FMVPOP      ; FR1 <- FRO ; POP Y OFF STACK INTO FRO
A897          SFMUL      ; X <- Y*X   FRO <- FRO * FR1
A897 20 DB DA   JSR    FMUL
A89A 4C B6 A9   JMP    CRYCHK

```

```

A89D          DSPM3
A89D C9 0A      CMP    #10      ; SET UP TO DISPLAY MEM REG A
A89F B0 05      BCS    DM10     ; ONLY DISPLAY 0-9
A8A1 1B         CLC
A8A2 69 05      ADC    #ROWREG  ; ROW = MEMNUM+4
A8A4 B5 54      STA    ROWCRS
A8A6          SXRTN
A8A6          ZMRTN
A8A6          DM10
A8A6 60         RTS

```

```

A8A7          SSUM
A8A7 20 D0 A3   JSR    MEMSUB    ; MEM <- MEM+X
A8AA 20 6A A9   JSR    SFADD
A8AD 20 65 A8   JSR    SSTO10
A8B0 4C 9D 9F   JMP    FPOPO     ; RELOAD X VALUE
A8B3          SXCHM
A8B3 20 D0 A3   JSR    MEMSUB    ; X <=> MEM (MEMSUB PUSHES FRO)
A8B6 20 C0 A3   JSR    MEMLDR    ; MEM <- X
A8B9 20 60 9F   JSR    FST1R
A8BC 20 E8 9F   JSR    SXCHGY    ; EXCHANGE NEW X FOR OLD ON STACK
A8BF 20 B6 9D   JSR    TOKNUM
A8C2 20 72 A8   JSR    DSPMEM    ; DISPLAY IN MEM AREA IN DSPFLG CLEAR
A8C5 4C 9D 9F   JMP    FPOPO
A8C8          SAND
A8C8 A9 00      LDA    #0
A8CA F0 06      BEQ    DOLOP     ; JMP
A8CC A9 01      SOR    LDA    #1  ; Y OR X
A8CE D0 02      BNE    DOLOP     ; JMP
A8D0          SXOR
A8D0 A9 02      LDA    #2
A8D2          DOLOP      ; X <- Y LOP X   0=>AND, 1=>OR, 2=>XOR
A8D2 B5 9E      STA    T0
A8D4 20 E6 9C   JSR    FPBIN
A8D7 A2 03      LDY    #3
A8D9          LOPLP1
A8D9 B5 B0      LDA    BINARY, X
A8DB 95 B4      STA    BIN2, X
A8DD CA         DEX
A8DE 10 F9      BPL    LOPLP1
A8E0 20 9D 9F   JSR    FPOPO
A8E3 20 E6 9C   JSR    FPBIN
A8E6 A4 9E      LDY    T0
A8E8 A2 03      LDY    #3
A8EA          LOPLP2
A8EA B5 B0      LDA    BINARY, X
A8EC C0 00      CPY    #0

```

A8EE D0 05		BNE	LOP10	
A8F0 35 B4		AND	BIN2, X	
A8F2 4C 00 A9		JMP	LOP30	
A8F5	LOP10			
A8F5 C0 01		CPY	#1	
A8F7 D0 05		BNE	LOP20	
A8F9 15 B4		ORA	BIN2, X	
A8FB 4C 00 A9		JMP	LOP30	
A8FE	LOP20			
A8FE 55 B4		EDR	BIN2, X	
A900	LOP30			
A900 95 B0		STA	BINARY, X	
A902 CA		DEX		
A903 10 E5		BPL	LOPLP2	
A905 4C 22 AB		JMP	BINFP	
A908	SPRINT			
A908 A5 95		LDA	PRNFLG	PRINT X REG
A90A D0 0B		BNE	SXR10	; PRINTER ON?
A90C 20 37 A7		JSR	SON	; YES.
A90F B0 95		BCS	SXRTN	; NO. TURN ON
A911 20 17 A9		JSR	SXR10	; ERROR
A914 4C 2E A7		JMP	SOFF	; DISPLAY & PRINT
A917	SXR10			; TURN OFF & RETURN
A917 20 7B A5		JSR	DSPCLR	DISPLAY & PRINT & RETURN
A91A 4B		PHA		; CLEAR DSPFLG
A91B 20 57 9D		JSR	FDSP0	; AND SAVE OLD VALUE.
A91E 4C 59 A5		JMP	DSPLOD	; PRINT NUMBER
				; RESTORE DSPFLG

COLLEEN CALCULATOR, BY C SHAW

A921		Z1IMN			; (1+I)^-N = 1/((1+I)^N)
A921 20 DB AD		JSR	Z1IN		
A924		SRECIP			
A924 A9 01		LDA	#1		
A926 20 B4 A1		JSR	LDINT		; COMPUTE A/FRO
A929 90 0F		BCC	SF DIV		; JMP
A92B		STAN			; TAN(X) = SIN(X)/COS(X)
A92B 20 BB 9F		JSR	FPUSHO		
A92E 20 6A B0		JSR	SSIN		
A931 20 E8 9F		JSR	SXCHGY		
A934 20 76 B2		JSR	SCOS		
A937		SPDIV			
A937 20 9A 9F		JSR	FMVPOP		
A93A		SF DIV			; X <- Y/X
A93A 20 28 DB		JSR	FDIV		
A93D 4C B6 A9		JMP	CRYCHK		
A940		SC			
A940 A9 5A		LDA	#CELMG		; DISPLAY "-> FAHRENHEIT"
A942 20 F0 9B		JSR	PUTMSG		
A945 A2 3C		LDX	#C1PT8		; CELSIUS -> FAHRENHEIT F=(9/5)*C+32
A947 20 C3 AD		JSR	LDY1ML		; FRO <- FRO*1.8
A94A A9 20		LDA	#32		
A94C D0 05		BNE	INTADD		; JMP
A94E		Z1PLI			; COMPUTE (1+I)
A94E 20 9E A3		JSR	LDI		
A951		ONEADD			
A951 A9 01		LDA	#1		; FRO <- FRO+1
A953		INTADD			
A953 20 B4 A1		JSR	LDINT		; COMPUTE A+FRO
A956 90 12		BCC	SFADD		; JMP
A958		SY			
A958 20 BB 9F		JSR	FPUSHO		; Y <- M*X + B
A95B 20 3A B0		JSR	SSLOPE		; SAVE X
A95E 20 94 AB		JSR	SPMUL		; M
A961 20 BB 9F		JSR	FPUSHO		
A964 20 1C B0		JSR	SYINTE		; B = Y-INTERCEPT
A967		SPADD			
A967 20 9A 9F		JSR	FMVPOP		
A96A		SFADD			; X <- Y+X
A96A 20 66 DA		JSR	FADD		
A96D 4C B6 A9		JMP	CRYCHK		
A970		Z11IMN			; 1-(1+I)^-N
A970 20 21 A9		JSR	Z1IMN		
A973 A9 01		LDASUB	#1		; FRO <- 1-FRO
A975		INTSUB			
A975 20 B4 A1		JSR	LDINT		; COMPUTE A-FRO
A978 90 09		BCC	SFSUB		; JMP
A97A		SFRACT			; FRO <- FRACTIONAL PART OF FRO
A97A 20 BB 9F		JSR	FPUSHO		
A97D 20 7D A6		JSR	STRUNC		; FRO-TRUNC(FRO)
A980		SPSUB			
A980 20 9A 9F		JSR	FMVPOP		
A983		SFSUB			; X <- Y-X
A983 20 60 DA		JSR	FSUB		
A986		CRYCHK			; IF CARRY SET THEN "ARITHMETIC OVERFLOW" ERROR
A986 90 43		BCC	SBST50		; RETURN IF CARRY CLEAR
A988		CRYSND			

A988 20 B0 A1
A98B A9 A9
A98D 4C B7 9B

JSR
LDA
JMP

PCLRO
#CRYMSG
ERRSUB

; CLEAR X
; CARRY SET => ERROR

SUBROUTINES FOR PROGRAMMABILITY

```

A990          SBSTEP          ; BACK STEP      PC <- PC-1
A990 A5 B9          LDA      PC
A992 38            SEC
A993 E9 01          SBC      #1
A995 B0 0C          BCS      SBST10
A997 A6 BA          LDX      PC+1
A999 CA            DEX
A99A E4 D1          CPX      PRGADR+1          ; AT BEGINNING OF PRGMEM?
A99C B0 03          BCS      SBST05
A99E 4C 60 9C       JMP      EPERR          ; YES. END OF PROGRAM MEM ERROR MSG AND RETURN
A9A1          SBST05
A9A1 86 BA          STX      PC+1
A9A3          SBST10
A9A3 85 B9          STA      PC
A9A5 A0 00          LDY      #0          ; CHECK FOR NUMBER
A9A7 B1 B9          LDA      (PC),Y
A9A9 C9 8E          CMP      #NUMBER
A9AB D0 1E          BNE      SBST50          ; RETURN
A9AD A5 B9          LDA      PC          ; NUMBER => SUBTRACT MORE
A9AF A6 BA          LDX      PC+1
A9B1 38            SEC
A9B2 E9 07          SBC      #FPREC+1
A9B4 B0 05          BCS      SBST30
A9B6 CA            DEX
A9B7 E4 D1          CPX      PRGADR+1          ; NOW AT BEGINNING OF PRGMEM?
A9B9 90 07          BCC      NERR
A9BB          SBST30
A9BB 48            PHA
A9BC B1 B9          LDA      (PC),Y          ; SAVE NEW PC
A9BE A8            TAY          ; DOUBLE CHECK FOR # AT BEGINNING
A9BF 68            PLA          ; RESTORE NEW PC
A9C0 C0 8E          CPY      #NUMBER
A9C2 F0 03          BEQ      SBST40          ; OK
A9C4          NERR
A9C4 4C B5 9B       JMP      KEYERR
A9C7          SBST40
A9C7 86 BA          STX      PC+1          ; SAVE NEW PC
A9C9 85 B9          STA      PC
A9CB          SBST50
A9CB 60            RTS

A9CC          SSSTEP          ; SINGLE STEP: IF IN STORE PROG MODE, THEN INC PC
; IF IN IMMEDIATE MODE, EXECUTE 1 INSTRUCTION
A9CC A5 BB          LDA      PROG
A9CE C9 01          CMP      #STOPRG
A9D0 F0 07          BEQ      SSTP10
A9D2 A9 02          LDA      #EXEC          ; IMMEDIATE -> EXECUTE MODE
A9D4 85 BE          STA      SSTFLG
A9D6 85 BB          STA      PROG
A9D8 60            RTS
A9D9          SSTP10
A9D9 A0 00          LDY      #0          ; CHECK FOR NUMBER
A9DB B1 B9          LDA      (PC),Y
A9DD C9 8E          CMP      #NUMBER
A9DF D0 11          BNE      SSTP20
A9E1 A0 07          LDY      #FPREC+1
A9E3 B1 B9          LDA      (PC),Y
A9E5 C9 8E          CMP      #NUMBER

```

COLLEEN CALCULATOR, BY C SHAW

A9E7 F0 06		BEQ	SSTP15	
A9E9 20 99 A1		JSR	PCINC	
A9EC 4C B5 9B		JMP	KEYERR	
A9EF	SSTP15			
A9EF 4C 9D A1		JMP	PCADDN	
A9F2	SSTP20			
A9F2 4C 99 A1		JMP	PCINC	
				; CLEAR PROGRAM MEMORY, PC <- PRGMEM
A9F5	SCLPRO			
A9F5 A9 00		LDA	#0	
A9F7 B5 B9		STA	PC	; OPCODE FOR STOP => INIT ALL TO STOP
A9F9 A9 6E		LDA	#STP	
A9FB A4 D1		LDY	PRGADR+1	
A9FD B4 BA		STY	PC+1	
A9FF A6 D3		LDX	PCIMX1	
AA01 D0 06		BNE	RAMSET	; JMP
				; CLEAR MEMORY REGISTERS
AA03	MEMCLR			; START ADDR MSB
AA03 A4 CF		LDY	MEMADR+1	
AA05 A6 D1		LDX	PRGADR+1	; END ADDR MSB
AA07	RAMCLR			; SET PAGES Y TO X-1 TO 0
AA07 A9 00		LDA	#0	; SET PAGES Y TO X-1 TO A
AA09 48	RAMSET	PHA		
AA0A B4 BD		STY	CLRPTR+1	; MEM UPPER LIMIT
AA0C B6 9E		STX	TO	
AA0E A9 00		LDA	#0	
AA10 B5 8C		STA	CLRPTR	
AA12 AB		TAY		
AA13 68		PLA		
AA14	INIT3			
AA14 91 BC		STA	(CLRPTR), Y	
AA16 C8		INY		
AA17 D0 FB		BNE	INIT3	
AA19 E6 BD		INC	CLRPTR+1	
AA1B A6 BD		LDX	CLRPTR+1	
AA1D E4 9E		CPX	TO	
AA1F D0 F3		BNE	INIT3	
AA21 60		RTS		; RETURN A= MEM CONTENTS

				; LIST PROGRAM STARTING WITH PC
AA22	SLIST			
AA22 20 BB 9F		JSR	FPUSH0	; SAVE X
AA25	SLSTLP			
AA25 20 5A 9C		JSR	DSPRG	
AA28 B0 1E		BCS	SLST10	
AA2A 20 9D A2		JSR	PUTCRP	
AA2D AD F0 02		LDA	CRSINH	; CURSOR_ON => BREAK HIT
AA30 F0 13		BEQ	BRKLSL	; BREAK
AA32 A5 B1		LDA	TOKCOD	; NO BREAK
AA34 C9 BE		CMP	#NUMBER	
AA36 D0 06		BNE	SLST05	
AA38 20 9D A1		JSR	PCADDN	
AA38 4C 41 AA		JMP	SLST07	
AA3E	SLST05			
AA3E 20 99 A1		JSR	PCINC	
AA41	SLST07			
AA41 90 E2		BCC	SLSTLP	; OK: CONTINUE

AA3E 20 01 A1	JSR	PCINC	
AA41	SLST07		
AA41 90 E2	BCC	SLSTLP	; OK: CONTINUE

COLLEEN CALCULATOR, BY C SHAW

AA43 B0 03	BCS	SLST10	; END OF PROG MEM: STOP
AA45	BRKLST		
AA45 EE F0 02	INC	CRSINH	; TURN CURSOR OFF
AA48	SLST10		
AA48 4C 9D 9F	JMP	FPOPO	; RESTORE X

AA4B	SPAUSE		
AA4B 20 7B A5	JSR	DSPCLR	
AA4E 4B	PHA		
AA4F 20 EB A1	JSR	DSOME	; DISPLAY REGS & STACK PAUSE FOR 30 FRAMES (1/2 SEC)

AA52 A2 00	LDX	#0	; MSB
AA54 A0 1E	LDY	#30	; LSB (IN FRAMES)
AA56 A9 03	LDA	#3	
AA5B 8D 2A 02	STA	CDTMF3	; SET FLAG TO NON-ZERO
AA5B 20 5C E4	JSR	SETVBV	; SET TIMER
AA5E	SPAULP		
AA5E AD 2A 02	LDA	CDTMF3	; WAIT FOR ZERO (TIME UP)
AA61 D0 FB	BNE	SPAULP	
AA63 4C 59 A5	JMP	DSPL0D	; RESTORE DSPFLG

AA66	SPROGR		; TO STORE PROGRAM MODE
AA66 A9 01	LDA	#STOPRG	
AA6B C5 BB	CMP	PROG	; ALREADY IN MODE?
AA6A F0 26	BEQ	SNOP	; YES. RETURN
AA6C B5 BB	STA	PROG	; NO.

AA6E A9 0A	LDA	#10	; CLEAR LINES
AA70 B5 9E	STA	TO	
AA72 A9 01	LDA	#LMARG	
AA74 B5 55	STA	COLCRS	
AA76 A9 02	LDA	#2	
AA7B B5 54	STA	ROWCRS	
AA7A A9 9C	LDA	#DELLIN	
AA7C 20 31 A2	JSR	PTCHR	
AA7F C6 9E	DEC	TO	
AAB1 10 F7	BPL	SPROLP	

AAB3 A2 5C	LDX	#FPX	; SAVE X REG
AAB5 A0 05	LDY	#FPX/256	
AAB7 4C A7 DD	JMP	ESTOR	

AAB8	SRESET		
AAB8 A9 00	LDA	#0	; PC ← 0
AAB8 B5 B9	STA	PC	
AABE A6 D1	LDX	PRGADR+1	
AA70 B6 BA	STX	PC+1	

AA92	SNOP		
AA92 60	RTS		

AA93	SSTP		; STOP PROGRAM EXECUTION
AA93	SEND		; END OF PROGRAM (STOP PROGRAM EXECUTION)
AA93 A9 00	LDA	#0	
AA95 B5 BD	STA	DSPFLG	; DISPLAY ON
AA97 A6 BB	LDX	PROG	

```

AA99 85 BB          STA      PRDG          ; BACK TO IMMEDIATE MODE
AA9B E0 01          CPX      #STOPRG       ; LEAVING STORE PROGRAM MODE?
AA9D D0 5F          BNE      DSCAL2        ; NO.
AA9F A2 5C          LDX      #FPX          ; YES. RELOAD FRO (X)
AAA1 A0 05          LDY      #FPX/256
AAA3 20 89 DD        JSR      FLDOR
AAA6                DSPALL
AAA6 A2 02          LDX      #2
AAA8 86 54          STX      ROWCRS
AAAA CA            DEX
AAAA 86 55          STX      COLCRS
AAAD CA            DEX
AAAE 20 49 A2        JSR      PTLIN1        ; LINE 2

; LINE 3  "I STACK  IREG . . .
AAB1 A9 F1          LDA      #STKLIN
AAB3 20 04 9C        JSR      STMSG2       ; SET UP MESSAGE IN TOKBUF
AAB6 20 74 A2        JSR      PUTCHS

AAB9 A2 03          LDX      #3            ; LINE 4
AAB3 20 49 A2        JSR      PTLIN1

AABE A2 00          LDX      #0            ; LINES 5-14  "IX  IO  OI"
AAC0                PTLP
AAC0 86 9E          STX      TO
AAC2 A9 7C          LDA      #'I
AAC4 20 31 A2        JSR      PTCHR
AAC7 A6 9E          LDX      TO
AAC9 BD FD BA        LDA      CHTAB2,X     ; X, Y, 2, 3, 4. . .
AACF 20 31 A2        JSR      PTCHR
AACF 20 6E A2        JSR      BLNK15
AAD2 A9 7C          LDA      #'I
AAD4 20 31 A2        JSR      PTCHR
AAD7 A5 9E          LDA      TO
AAD9 18            CLC
AADA 69 30          ADC      #'0
AADC 20 31 A2        JSR      PTCHR       ; '0 - '9
AADF A2 12          LDX      #1B
AAE1 20 70 A2        JSR      BLNKS
AAE4 A9 7C          LDA      #'I
AAE6 20 31 A2        JSR      PTCHR
AAE9 A6 9E          LDX      TO
AAEB EB            INX
AAEC E0 0A          CPX      #10
AAEE D0 D0          BNE      PTLP

; WIDEN MARGINS SO LOGICAL LINES COME OUT RIGHT
AAFO E6 53          INC      RMARGN
AAF2 A2 06          LDX      #6            ; LINE 15
AAF4 20 49 A2        JSR      PTLIN1
AAF7 20 0B A2        JSR      PUTCR
AAFA C6 53          DEC      RMARGN
AAFC D0 05          BNE      DSCAL        ; JMP
AAFE                DSCAL2
AAFE A9 40          LDA      #$40
AB00 20 3A 9C        JSR      SOUND        ; STOP PROGRAM SOUND
AB03                DSCAL
AB03 4C EB A1        JMP      DSOME        ; DISPLAY STACK, MEM, & X

```


AB06		XLTSUB		; XLT R N => IF X < MEM(R) THEN GOTO N
				SUBROUTINE FOR CONDITIONAL BRANCH INSTRUCTIONS
AB06 20 D0 A3		JSR	MEMSUB	; WILL RETURN FROM XLTSUB IF ERROR, OTHERWISE
AB09 20 B3 A9		JSR	SFSUB	; FRO <- MEM(R) - X
AB0C A5 D4		LDA	FRO	; LOAD & SAVE SIGN BYTE
AB0E 48		PHA		
AB0F 20 9D 9F		JSR	FPOPO	; RELOAD X
AB12 68		PLA		
AB13 18		CLC		; NO ERROR
AB14				
AB14 60	XLTERR	RTS		
AB15				
AB15 20 06 AB	SXEQ	JSR	XLTSUB	; IF X=MEM(R) THEN GOTO N
AB18 B0 FA		BCS	XLTERR	; ERROR
AB1A F0 24		BEQ	MATCH	
AB1C				
AB1C 20 54 AB	NOMAT	JSR	SG01	; CONDITION NOT SATISFIED
AB1F B0 F3		BCS	XLTERR	; CALL LEX AND CHECK FOR # IN RANGE
AB21 90 2C		BCC	SG02	; ERROR => RETURN
AB23	SXGE			; NO ERROR => RESTORE X & DHOF LG & RETURN
AB23 20 06 AB		JSR	XLTSUB	; IF X>= MEM(R) THEN GOTO N
AB26 B0 EC		BCS	XLTERR	
AB28 F0 16		BEQ	MATCH	
AB2A 30 14		BMI	MATCH	
AB2C 10 EE		BPL	NOMAT	; MI => MEM(R) < X => X>MEM(R)
AB2E	SXLT			; IF X<MEM(R) THEN GOTO N
AB2E 20 06 AB		JSR	XLTSUB	
AB31 B0 E1		BCS	XLTERR	
AB33 F0 E7		BEQ	NOMAT	
AB35 10 09		BPL	MATCH	; PL => MEM(R)>=X => X<=MEM(R)
AB37 30 E3		BMI	NOMAT	
AB39	SXNE			; IF X<>MEM(R) THEN GOTO N
AB39 20 06 AB		JSR	XLTSUB	
AB3C B0 D6		BCS	XLTERR	
AB3E F0 DC		BEQ	NOMAT	
AB40	MATCH			
AB40	SGOTO			; GOTO N = 0-1023 (000-3FF)
AB40 20 54 AB		JSR	SG01	; CALL LEX & CHECK FOR NUMBER IN RANGE
AB43 B0 CF		BCS	XLTERR	; ERROR => RETURN
AB45 A6 D4		LDX	FRO	
AB47 B6 B9		STX	PC	
AB49 A5 D5		LDA	FRO+1	
AB4B 65 D1		ADC	PRGADR+1	
AB4D B5 BA		STA	PC+1	
AB4F	SG02			; ENTRY POINT
AB4F 20 9D 9F		JSR	FPOPO	; RELOAD X
AB52 18		CLC		; NO ERROR
AB53 60		RTS		
AB54				
AB54 20 BB 9F	SG01	JSR	FPUSH0	; SAVE X
AB57 A9 00		LDA	#PROMSG	; DISPLAY "ENTER PROGRAM MEM ADDRESS 0-1023"
AB59 20 F0 9B		JSR	PUTMSG	
AB5C A5 87		LDA	DHOF LG	; ALWAYS DECIMAL
AB5E 48		PHA		
AB5F A9 00		LDA	#0	
AB61 B5 87		STA	DHOF LG	

COLLEEN CALCULATOR, BY C SHAW

AB63 20 51 9A	JSR	LEX	
AB66 A5 81	LDA	TOKCOD	
AB68 C9 8E	CMP	#NUMBER	
AB6A D0 0F	BNE	SGOERR	
AB6C 20 D2 D9	JSR	FPI	
AB6F B0 0A	BCS	SGOERR	
AB71 A5 D5	LDA	FRO+1	
AB73 C9 04	CMP	#4	
AB75 B0 04	BCS	SGOERR	; TOO LARGE
AB77 68	PLA		
AB7B 85 87	STA	DHOFLO	; RESTORE DHOFLO
AB7A 60	RTS		

AB7B	SGOERR		
AB7B 68	PLA		
AB7C 85 87	STA	DHOFLO	
AB7E 20 9D 9F	JSR	FPOPO	; RELOAD X
AB81 4C 95 A3	JMP	BITERR	

SUBROUTINE CALL & RETURN

```

;
AB84 A2 00      SCLCAL LDX      #0          ; CLEAR CALL STACK
AB86 F0 14      BEQ      PSHC20        ; JMP
AB88 20 D6 AB    SPOPC  JSR      POPCAL     ; POP CALL STACK (TWO POPS)
AB8B 90 49      BCC      POPCAL
AB8D 60          RTS

AB8E            PSHCAL                    ; SAVE A ON CALSTK
AB8E 18          CLC                      ; NO ERROR
AB8F A6 C9      LDX      CALPTR
AB91 10 05      BPL      PSHC10
AB93 A9 CA      LDA      #CLFMSG        ; "CALL STACK FULL" ERROR
AB95 4C B7 9B    ERRCAL JMP      ERRSUB
AB98            PSHC10
AB98 9D 80 04    STA      CALSTK, X
AB9B E8          INX
AB9C            PSHC20
AB9C 86 C9      STX      CALPTR
AB9E 60          RTS

AB9F            SCALL                    ; CALL N (0-1023)
AB9F A6 BB      LDX      PROG           ; IMMEDIATE MODE?
ABA1 D0 16      BNE      SCAL10        ; NO.
ABA3 A5 BA      LDA      PC+1          ; YES. PC TO BE RESTORED ON RETURN
ABA5 09 80      DRA      #$80          ; SET MSBIT TO INDICATE RETURN TO IMMEDIATE MODE
ABA7 20 8E AB    JSR      PSHCAL
ABAA A5 B9      LDA      PC
ABAC 20 8E AB    JSR      PSHCAL
ABAF B0 4C      BCS      SRET20        ; STACK FULL ERROR
ABB1 20 40 AB    JSR      SGOTO        ; GOTO N
ABB4 B0 1D      BCS      SCAL30        ; ERROR
ABB6 4C F0 BC    JMP      SCONTI       ; RUN SUBROUTINE (CONTINUE)
ABB9            SCAL10
ABB9 A5 B9      LDA      PC
ABBB A6 BA      LDX      PC+1          ; EXEC MODE
ABBD 18          CLC                      ; SKIP PAST N
ABBE 69 08      ADC      #FPREC+2      ; TO GET RETURN ADDRESS.
ABC0 90 01      BCC      SCAL20
ABC2 E8          INX
ABC3            SCAL20
ABC3 48          PHA
ABC4 8A          TXA
ABC5 20 8E AB    JSR      PSHCAL        ; PC+1
ABC8 68          PLA
ABC9 20 8E AB    JSR      PSHCAL        ; PC
ABCC B0 2F      BCS      SRET20        ; STACK FULL ERROR => DON'T GO
ABCE 20 40 AB    JSR      SGOTO
ABD1 90 2A      BCC      SRET20
ABD3            SCAL30
ABD3 20 D6 AB    JSR      POPCAL        ; ERROR => THROW AWAY RETURN ADDR FROM STACK
;                                     CALL POPCAL A 2ND TIME

ABD6            POPCAL                    ; POP A OFF CALSTK
ABD6 18          CLC                      ; NO ERROR
ABD7 A6 C9      LDX      CALPTR
ABD9 D0 04      BNE      POPC10
ABDB A9 C0      LDA      #CLEMSG
ABDD D0 B6      BNE      ERRCAL        ; JMP "CALL STACK EMPTY" ERROR
ABDF            POPC10

```

COLLEEN CALCULATOR, BY C SHAW

ABDF CA
 ABE0 B6 C9
 ABE2 BD 80 04
 ABE5 60

DEX
 STX CALPTR
 LDA CALSTK, X
 RTS

ABE6
 ABE6 20 D6 AB
 ABE9 80 12
 ABE8 85 B9
 ABED 20 D6 AB
 ABF0 80 0B
 ABF2 10 07
 ABF4 29 7F
 ABF6 85 BA
 ABF8 4C 93 AA
 ABFB
 ABFB 85 BA
 ABFD
 ABFD 60

SRETUR

SRET10

SRET20

JSR POPCAL
 BCS SRET20
 STA PC
 JSR POPCAL
 BCS SRET20
 BPL SRET10
 AND #7F
 STA PC+1
 JMP SSTP

 STA PC+1
 RTS

; RETURN => POP PC OFF STACK, GOTO PC

; PC

; ERROR - STACK EMPTY

; PC+1

; STACK EMPTY => DON'T EXECUTE RETURN

; PC+1 (MSB) <0 => RETURN TO IMMEDIATE MODE

; PC+1 >0 => STAY IN EXEC MODE

INSERT & DELETE

ABFE	SDELET			; DELETE - FOR I=PC TO 1022+PRGMEM: MEM(I)<-MEM(I+1): NEXT I
ABFE 20 61 A1		JSR	NCHKLD	MEM(1023+PRGMEM)<-STP
AC01 B0 0D		BCS	SDEL2	; NUMBER?
AC03 D0 0B		BNE	SDEL2	; NO. ERROR DELETE 1 BYTE
AC05 A9 06		LDA	#FPREC	; NO. DELETE 1 BYTE
AC07 B5 9E		STA	TO	; YES. DELETE 8 BYTES FOR NUMBER
AC09	SDELP2			
AC09 20 10 AC		JSR	SDEL2	
AC0C C6 9E		DEC	TO	
AC0E 10 F9		BPL	SDELP2	; 7 TIMES
				8TH CALL

AC10	SDEL2			; DELETE 1 BYTE FROM PRGMEM
AC10 A5 B9		LDA	PC	; MOVE PC TO TEMP PTR
AC12 B5 90		STA	JMPTR1	
AC14 A5 BA		LDA	PC+1	
AC16 B5 91		STA	JMPTR1+1	
AC18 A0 01	SDELP1	LDY	#1	
AC1A B1 90		LDA	(JMPTR1), Y	; MEM(I+1)
AC1C B8		DEY		
AC1D 91 90		STA	(JMPTR1), Y	; MEM(I)
AC1F E6 90		INC	JMPTR1	
AC21 D0 F5		BNE	SDELP1	; CONTINUE
AC23 E6 91		INC	JMPTR1+1	
AC25 A5 91		LDA	JMPTR1+1	
AC27 C5 D3		CMP	PC1MX1	; AT END OF MEM?
AC29 D0 ED		BNE	SDELP1	; NO. CONTINUE
AC2B A9 6E		LDA	#STP	; DONE STORE "STOP" AT END OF PRGMEM
AC2D C6 91		DEC	JMPTR1+1	JMPTR1 = 0
AC2F A0 FF		LDY	##FF	; PC1MAX
AC31 91 90		STA	(JMPTR1), Y	
AC33 60		RTS		

AC34	SINSER			; INSERT - FOR I=1022+PRGMEM TO PC: MEM(I+1)<-MEM(I): NEXT I
AC34 A9 FE		LDA	##FE	MEM(PC)<-STP
AC36 B5 90		STA	JMPTR1	JMPTR1<-ADDR(END OF PRGMEM-1)
AC38 A5 D2		LDA	PC1MAX	
AC3A B5 91		STA	JMPTR1+1	
AC3C	SINSLP			
AC3C A0 00		LDY	#0	
AC3E B1 90		LDA	(JMPTR1), Y	; MEM(I)
AC40 C8		INY		
AC41 91 90		STA	(JMPTR1), Y	; MEM(I+1)
AC43 C6 90		DEC	JMPTR1	
AC45 A6 90		LDX	JMPTR1	
AC47 E0 FF		CPX	##FF	
AC49 D0 02		BNE	INS10	
AC4B C6 91		DEC	JMPTR1+1	
AC4D	INS10			
AC4D A4 91		LDY	JMPTR1+1	
AC4F C4 BA		CPY	PC+1	
AC51 90 06		BCC	INS30	; JMPTR1<PC => STOP
AC53 D0 E7		BNE	SINSLP	; JMPTR1>PC => CONTINUE
AC55 E4 B9		CPX	PC	; JMPTR1+1 = PC+1

COLLEEN CALCULATOR, BY C SHAW

AC57 B0 E3		BCS	SINSLP	; JMPTR1 >= PC => CONTINUE
AC59	INS30			
AC59 A0 00		LDY	#0	; DONE MEM(PC) <- STP
AC5B A9 6E		LDA	#STP	
AC5D 91 B9		STA	(PC), Y	
AC5F 60		RTS		

SAVE & LOAD

```

;
AC60 48          FOPEN  PHA          ; OPEN FILE FOR INPUT/OUTPUT (ACCORDING TO A)
AC61 A9 38        LDA          ; "ENTER FILESPEC" MESSAGE
AC63 20 F0 9B     JSR          PUTMSG
AC66             FOPLP1  JSR          LXINIT      ; SET UP CURSOR, DISPLAY '>'
AC66 20 3C 9A     JSR          GTCHR      ; REMOVE LEADING SEPARATORS
AC67 20 26 A0     CMP          #'
AC6C C9 20        BEQ          FOPLP1
AC6E F0 F6        CMP          #DELLIN    ; DELETE LINE?
AC70 C9 9C        BEQ          FOPLP1      ; YES. START OVER
AC72 F0 F2
AC74             FOP20   INC          TOKPTR    ; SAVE CHAR
AC74 E6 82        JSR          GTCHR
AC76 20 26 A0     CMP          #'
AC79 C9 20        BEQ          FOP30      ; DONE
AC7B F0 10        CMP          #DELLIN
AC7D C9 9C        BEQ          FOPLP1      ; TRY AGAIN
AC7F F0 E5        LDX          TOKPTR
AC81 A6 82        CPX          #NUMLEN
AC83 E0 0E        BCC          FOP20
AC85 90 ED        PLA
AC87 68           LDA          #DIGMSG
AC88 A9 9D        JMP          ERRSUB      ; "TOO MANY CHARS" ERROR MESSAGE
AC8A 4C B7 9B
AC8D             FOP30   INC          CRSINH
AC8D EE F0 02     JSR          PUTCR      ; CURSOR OFF (INHIBIT ON)
AC90 20 0B A2     LDA          #CR        ; PUT CR SO >FILESPEC WILL BE DISPLAYED
AC93 A9 9B        LDX          TOKPTR
AC95 A6 82        STA          TOKBUF, X   ; PUT CR AT END OF STRING
AC97 9D 00 05     LDX          #TIOCB     ; TEMPORARY IOCB #
AC9A A2 30        PLA                ; INPUT OR OUTPUT
AC9C 68           STA          ICAX1, X
AC9D 9D 4A 03     LDY          #8        ; OPEN
ACA0 A0 08        CIOCAL JSR          CIOIN2   ; SET UP IOCB X, CALL CIO, AND CHECK FOR SUCCESS
ACA2 20 FC AC     BNE          IOERR      ; ERROR
ACA5 D0 02        CLC                ; NO ERROR
ACA7 18           RTS
ACAB 60

ACA9             IOERR  TYA
ACA9 9B           IOERR2
ACAA             IOERR2 PHA
ACAA 4B           JSR          ERRSB2      ; SAVE
ACAB 20 D7 9B     JSR          FPUSHO     ; DISPLAY "ERROR - ", DO OTHER STUFF
ACAE 20 BB 9F     PLA                ; SAVE X
ACB1 68           JSR          PSETO      ; CONVERT ERROR # TO FP TO ASCII (0-255)
ACB2 20 B9 A1     LDA          DHOFLG    ; FRO <- FP (A)
ACB5 A5 87        PHA                ; SAVE DHOFLG AND SET TO DECIMAL
ACB7 4B           LDA          #0
ACB8 A9 00        STA          DHOFLG
ACBA B5 87        JSR          FDSPO      ; DISPLAY ERROR NUMBER (WILL BE DECIMAL IN CURRENT FIX)
ACBC 20 57 9D     PLA
ACBF 68           STA          DHOFLG
ACCO B5 87

ACC2 20 9D 9F     JSR          FPOPO      ; RELOAD X
ACC5 38           SEC                ; ERROR
ACC6 60           RTS

```

```

ACC7          SLOAD          LDA    #INPUT          ;LOAD PROGRAM MEM FROM SPECIFIED FILE
ACC7 A9 04     JSR    FOPEN
ACC9 20 60 AC  BCS    FCLOSE          ;ERROR IF CRY SET
ACCC 80 22     LDA    #GETCHR
ACCE A9 07     JSR    SAVL0D          ;LOAD DATA
ACD0 20 14 AD  BNE    SLD10          ;STATUS NOT SUCCESS
ACD3 D0 05     JSR    EPERR          ;IN THIS CASE SUCCESS => ERROR BECAUSE FILE
ACD5 20 60 9C          BCS    FCLOSE          ;IS TOO LONG. DISPLAY "END OF PROGRAM MEM"
                                ;JMP TO CLOSE FILE
ACD8 B0 16     SLD10          CPY    #EOF          ;FILE JUST RIGHT IF END-OF-FILE
ACDA C0 03     JMP    SSAV10
ACDC 4C EB AC

```

```

ACDF A9 08     SSAVE          LDA    #OUTPUT          ;SAVE PROGRAM MEM IN FILE
ACE1 20 60 AC  JSR    FOPEN
ACE4 B0 0A     BCS    FCLOSE
ACE6 A9 08     LDA    #PUTCHR
ACE8 20 14 AD  JSR    SAVL0D          ;SAVE DATA
ACEB          SSAV10
ACEB F0 03     BEQ    FCLOSE          ;NO ERROR
ACED 20 A9 AC  JSR    IOERR          ;DISPLAY ERROR AND CLOSE
ACF0          FCLOSE          LDX    #TIOCB          ;CLOSE TIOCB
ACF0 A2 30     XCLOSE
ACF2          LDY    #11
ACF2 A0 08     BNE    CIOCAL          ;CLOSE
ACF4 D0 AC          ;JMP TO CIO CALL

```

```

ACF6          CIOINT          ;SET UP IOCB, CALL CIO, & CHECK FOR SUCCESS
;                                     INPUT: X=IOCB OFFSET, Y=CIOTAB OFFSET
;                                     OUTPUT: EQ=>SUCCESS, NE=>ERROR

```

```

ACF6 B9 0A BB  LDA    CIOTAB+3, Y
ACF7 9D 4A 03  STA    ICAX1, X
ACFC          CIOIN2
ACFC B9 09 BB  LDA    CIOTAB+2, Y
ACFF 9D 45 03  STA    ICBAH, X
AD02 B9 08 BB  LDA    CIOTAB+1, Y
AD05 9D 44 03  STA    ICBAL, X
AD08 B9 07 BB  LDA    CIOTAB+0, Y
AD0B 9D 42 03  STA    ICCOM, X
AD0E          CIOIN3
AD0E 20 56 E4  JSR    CIOV
AD11 C0 01     CPY    #SUCCESS
AD13 60          RTS

```

```

AD14          SAVL0D          ;SUBROUTINE TO LOAD OR SAVE DATA
AD14 A2 30     LDX    #TIOCB
AD16 9D 42 03  STA    ICCOM, X
AD17 A5 D1     LDA    PRGADR+1
AD1B 9D 45 03  STA    ICBAH, X
AD1E A9 00     LDA    #PRGLEN          ;0
AD20 9D 4B 03  STA    ICBLL, X
AD23 9D 44 03  STA    ICBAL, X          ;0

```

```

AD26 A9 04     LDA    #PRGLEN/256
AD28 9D 49 03  STA    ICBLL, X
AD2B D0 E1     BNE    CIOIN3          ;CALL CIO AND CHECK FOR ERROR

```


AD17 A2 D1	LDA	PRGADR1	
AD18 9D 45 03	STA	ICBAH, X	
AD1E A9 00	LDA	#PRGLEN	; 0
AD20 9D 48 03	STA	ICBLL, X	
AD23 9D 44 03	STA	ICBAL, X	; 0

COLLEEN CALCULATOR, BY C SHAW

AD26 A9 04	LDA	#PRGLEN/256	
AD28 9D 49 03	STA	ICBLH, X	
AD2B D0 E1	BNE	CIOIN3	; CALL CIO AND CHECK FOR ERROR JMP

SUBROUTINES FOR CONVERSIONS

AD2D				
AD2D A2 42	SM	LDX	#0*FPREC+LENGTH	; LOAD LSB OF ADDR OF CONVERSION CONSTANT
AD2F D0 16		BNE	LENG	; JMP
AD31	SIN			
AD31 A2 48		LDX	#1*FPREC+LENGTH	
AD33 D0 12		BNE	LENG	
AD35	SFT			
AD35 A2 4E		LDX	#2*FPREC+LENGTH	
AD37 D0 0E		BNE	LENG	
AD39 A2 54	SYD	LDX	#3*FPREC+LENGTH	
AD3B D0 0A		BNE	LENG	
AD3D A2 5A	SMI	LDX	#4*FPREC+LENGTH	
AD3F D0 06		BNE	LENG	
AD41 A2 60	SCM	LDX	#5*FPREC+LENGTH	
AD43 D0 02		BNE	LENG	
AD45 A2 66	SKM	LDX	#6*FPREC+LENGTH	
		BNE	LENG	
	SNAUTI	LDX	#7*FPREC+LENGTH	
AD47	LENG			
AD47 A9 A2		LDA	#ZM	; LSB OF MESSAGE ADDR
AD49 D0 2E		BNE	CONVRT	; JMP
AD4B	SKG			
AD4B A2 42		LDX	#LENGTH	; CONSTANT 1 (NO CONVERSION)
AD4D D0 0A		BNE	MAS	; JMP
AD4F A2 6C	SDZ	LDX	#0*FPREC+MASS	
AD51 D0 06		BNE	MAS	
AD53	SLB			
AD53 A2 72		LDX	#1*FPREC+MASS	
AD55 D0 02		BNE	MAS	
AD57	SGM			
AD57 A2 7B		LDX	#2*FPREC+MASS	
AD59	MAS			
AD59 A9 A5		LDA	#ZKG	
AD5B D0 1C		BNE	CONVRT	; JMP
AD5D A2 42	SFLOZ	LDX	#LENGTH	
AD5F D0 16		BNE	VOL	
AD61 A2 7E	STSP	LDX	#0*FPREC+VOLUME	
AD63 D0 12		BNE	VOL	
AD65 A2 84	STBSP	LDX	#1*FPREC+VOLUME	
AD67 D0 0E		BNE	VOL	
AD69 A2 8A	SCUP	LDX	#2*FPREC+VOLUME	
AD6B D0 0A		BNE	VOL	
AD6D A2 90	SGT	LDX	#3*FPREC+VOLUME	
AD6F D0 06		BNE	VOL	
AD71 A2 96	SGAL	LDX	#4*FPREC+VOLUME	
AD73 D0 02		BNE	VOL	
AD75 A2 9C	SL	LDX	#5*FPREC+VOLUME	
AD77	VOL			
AD77 A9 AA		LDA	#ZFL	
		IF	ASMBL	
		BNE	CONVRT	; JMP
	SCDEG	LDX	#LENGTH	; 1
		BNE	CDGR	; JMP
	SCGRAD	LDX	#0*FPREC+DEGREE	
		BNE	CDGR	
	SCRAD	LDX	#1*FPREC+DEGREE	
	CDGR	LDA	#ZDEG	; DEGREES MSG

SCDEG	LDX	#LENGTH	; 1
	BNE	CDGR	; JMP
SCGRAD	LDX	#0*FPREC+DEGREE	
	BNE	CDGR	
SCRAD	LDX	#1*FPREC+DEGREE	
CDGR	LDA	#ZDEG	; DEGREES MSG

COLLEEN CALCULATOR, BY C SHAW

.ENDIF			
AD79	CONVRT		; CONVERT TO DIFFERENT UNITS
AD79 A4 CA	LDY	CONFLG	; FLAG SET?
AD7B D0 17	BNE	CONV10	; YES.
AD7D B5 CA	STA	CONFLG	; NO. SAVE MSG ADDR LSB & DO INTERMEDIATE CONVERSION
AD7F A0 BA	LDY	#LENGTH/256	; LOAD CONVERSION CONSTANT
ADB1 20 98 DD	JSR	FLDIR	
ADB4 20 97 AB	JSR	SFMUL	; AND MULTIPLY.
ADB7 A5 CA	LDA	CONFLG	; DISPLAY NEW UNITS
ADB9 20 F0 93	JSR	PUTMSG	
ADB8 20 EE A1	JSR	FDSCOM	; DISPLAY NEW X
ADB8 A9 44	LDA	#CN2MSG	; DISPLAY "ENTER NEW UNITS"
AD91 4C F0 9B	JMP	PUTMSG	
AD94	CONV10		; CONVERT FROM INTERMEDIATE UNITS TO FINAL UNITS
AD94 C5 CA	CMP	CONFLG	; FLAGS MATCH?
AD96 D0 10	BNE	CONERR	; NO. ERROR - CAN'T MIX TYPES
AD98 A0 BA	LDY	#LENGTH/256	; YES. LOAD CONVERSION CONSTANT
AD9A 20 98 DD	JSR	FLDIR	
AD9D 20 3A A9	JSR	SFDIV	; AND DIVIDE.
ADA0 20 EE A1	JSR	FDSCOM	; DISPLAY NEW VALUE
ADA3 A9 4F	LDA	#CN3MSG	; DISPLAY "CONVERSION COMPLETE"
ADA5 4C F0 9B	JMP	PUTMSG	
ADAB	CONERR		
ADAB A9 D4	LDA	#UNIMSG	
ADAA 4C B7 9B	JMP	ERRSUB	; DISPLAY "UNIT MISMATCH" ERROR MESSAGE & RETURN
.IF ASMBL			
; NEW VERSION			
SM	LDA	#ZM	; METERS->FT
	JSR	PUTMSG	
	LDX	#CFT	
	BNE	LC1DIV	; JMP
SFT	LDA	#ZFT	; FT->METERS
	JSR	PUTMSG	
	LDX	#CFT	
	BNE	LC1MUL	; JMP
SLB	LDA	#ZLB	; LB->KG
	JSR	PUTMSG	
	LDX	#CLB	
	BNE	LC1MUL	
SKG	LDA	#ZKG	
	JSR	PUTMSG	; KG->LB
	LDX	#CLB	
	BNE	LC1DIV	
SCAL	LDA	#ZGAL	; GAL->LITERS (L)
	JSR	PUTMSG	
	LDX	#CL	
	BNE	LC1DIV	
SL	LDA	#ZLIT	; L->GAL
	JSR	PUTMSG	
	LDX	#CL	
	BNE	LC1MUL	
.ENDIF			

ADAD A9 9D	SCRAD	LDA	#ZRAD	; RAD->DEG
------------	-------	-----	-------	------------

COLLEEN CALCULATOR. BY C SHAW

ADAF 20 F0 9B		JSR	PUTMSG	
AD82 A2 36		LDX	#PIOV18	;PI/180
AD84 A0 BA	LDY1DV	LDY	#PICONST/256	;FRO <- FRO / (X,Y=PICONST/256)
AD86	LD1DIV			;FRO <- FRO / (X,Y)
AD86 20 98 DD		JSR	FLD1R	
AD89 4C 3A A9		JMP	SFDIV	
ADBC A9 99	SCDEG	LDA	#ZDEG	
ADBE 20 F0 9B		JSR	PUTMSG	
ADC1 A2 36		LDX	#PIOV18	
ADC3 A0 BA	LDY1ML	LDY	#PICONST/256	;FRO <- FRO * DATA CONSTANT (LSB OF ADDR IN X)
ADC5	LD1MUL			;FRO <- FRO * DATA CONSTANT (ADDR IN X & Y)
ADC5 20 98 DD		JSR	FLD1R	
ADC8 4C 97 AB		JMP	SFMUL	
	IF	ASMBL		
SMI	LDA	#ZMI		;MILES->KG
	JSR	PUTMSG		
	LDX	#CMI		
SKM				;KM->MI
	LDA	#ZKM		
	JSR	PUTMSG		
	LDX	#CMI		
	ENDIF			
ADCB	SF			
ADCB A9 5E		LDA	#FAHMSG	; DISPLAY "->CELSIUS"
ADCD 20 F0 9B		JSR	PUTMSG	
ADDO A9 20		LDA	#32	
ADD2 20 BF A6		JSR	SUBINT	;FRO <- FRO-32
ADD5 A2 3C		LDX	#C1PT8	;FAHRENHEIT -> CELSIUS C=(5/9)*(F-32)
ADD7 A0 BA		LDY	#C1PT8/256	
ADD9 D0 DB		BNE	LD1DIV	;JMP

COMPOUND INTEREST SUBROUTINES

```

;
SUBROUTINES TO COMPUTE PARTS OF COMPOUND INT. EQUATIONS
; COMPUTE (1+I)^N
Z1ILDN JSR
SPOWER JMP
Z1INM1 JSR
SUBONE JMP
Z11IMN JSR
FPU SHO JSR
LDI JSR
SPDIV JMP

ADDB 20 A2 A3 Z1IN JSR
ADDE 4C 19 A6 Z1INM1 JSR
ADE1 20 DB AD Z1INM1 JSR
ADE4 4C BD A6 Z11IMN JSR
ADE7 20 70 A9 Z11IMN JSR
ADEA 20 BB 9F DIVI JSR
ADED 20 9E A3 DIVI JSR
ADF0 4C 37 A9 DIVI JMP

; COMPOUND INTEREST
; JMP
; ANNUITY DUE (PAY AT BEGINNING OF PERIOD
; E.G. SAVINGS ACCT.)
; ORDINARY ANNUITY (PAY AT END OF PERIOD
; E.G. LOAN
; DISPLAY STATUS
; JMP TO CHSTAT

ADF3 A9 B0 SCMPND LDA #0
ADF5 D0 06 BNE SORD10
ADF7 A9 00 SFVDUE LDA #0
ADF9 F0 02 BEQ SORD10
ADFB A9 01 SFVORD LDA #1
ADFD B5 C5 SORD10 STA DUEFLQ
ADFF A9 1B LDA #DFVDUE
AE01 D0 12 BNE SFND20

; ANNUITY DUE/PV
; ORDINARY ANNUITY/PV

AE03 A9 02 SPVDUE LDA #2
AE05 D0 F6 BNE SORD10
AE07 A9 03 SPVORD LDA #3
AE09 D0 F2 BNE SORD10

; ENTER VALUE
; FIND VALUE, GIVEN OTHER VARIABLES
; DISPLAY STATUS

AE0B A9 00 SENTER LDA #0
AE0D F0 02 BEQ SFND10
AE0F A9 01 SFIND LDA #1
AE11 B5 C6 SFND10 STA ENTFLQ
AE13 A9 21 LDA #DENTER
AE15 SFND20
AE15 4C 5D A7 JMP CHSTAT

```

COLLEEN CALCULATOR, BY C SHAW

```

AE1B          ;BAL = BALLOON PAYMENT
AE1B A5 C6    ;BAL
AE1A F0 3C    LDA    ENTFLG
AE1C A5 C5    BEQ    SBAL05
AE1E 10 03    LDA    DUEFLQ
AE20          BPL    SBAL20
AE20 4C B5 9B ;SBAL15    JMP    KEYERR
AE23          SBAL20
AE23 29 02    AND    #2
AE25 F0 F9    BEG    SBAL15
; NO BAL IF FV OR CMPNDINTRST
BAL = (PV - PMT*(1-(1+I)^-N)/I)/(1+I)^-N
BAL = (PV - PMT*(1+I)*(1-(1+I)^-N)/I)/(1+I)^-N
; FRO <- PMT
; IF ANNUITY DUE THEN FRO <- FRO * (1+I)
AE27 20 AC A3 ;JSR    ANNUITY DUE:
AE2A 20 B9 AB ;JSR    LDPMT
AE2D 20 BB 9F ;JSR    ZMUL11
AE30 20 E7 AD ;JSR    FPUSHO
AE33 20 94 AS ;JSR    Z11INI
AE36 20 B6 DD ;JSR    SPMUL
AE39 20 B0 A3 ;JSR    FMOVE
AE3C 20 B3 A9 ;JSR    LDPV
AE3F 20 BB 9F ;JSR    SFSUB
AE42 20 21 A9 ;JSR    FPUSHO
AE45 20 37 A9 ;JSR    Z1IMN
AE48          ;JSR    SPDIV
AE48          ;PV
AE48 A9 04    ;B1+I)^-N
AE4A          ;ENTER VALUE
AE4A          SBAL05
AE4A          LDA    #4
AE4A 85 A3    MEMSTO
AE4C A9 02    STA    MEMNUM
AE4E 20 6B A5 ;LDA    #2
AE51 4C 65 AB ;JSR    SFIX2
; FIX 2 (DISPLAYING DOLLAR VALUE)
; STORE REG & DISPLAY
; JMP    SST010

AE54          SI
AE54 A5 C6    LDA    ENTFLG
AE56 D0 21    BNE    SI10
AE58 20 BB 9F ;JSR    FPUSHO
AE5B A9 64    LDA    #100
; CONVERT INTEREST IN PERCENT TO FRACTIONAL VALUE
; BY DIVIDING BY 100.
AE5D 20 B9 A1 ;JSR    PSET0
AE60 20 37 A9 ;JSR    SPDIV
AE63 A9 0B    SI05
AE65 20 6B A5 ;LDA    #8
AE68 A9 06    ;JSR    SFIX2
AE6A 85 A3    LDA    #6
; FIX 8 FOR I/100
AE6C 20 65 AB ;LDA    MEMNUM
AE6F A9 64    ;JSR    SFIX2
AE71 20 84 AB ;LDA    #100
AE74 A9 04    ;JSR    INTMUL
AE76 4C 6B A5 ;LDA    #4
AE79          ;JSR    SFIX2
; FIX 4 FOR I IN PERCENT (DISPLAY)
; AND RETURN
AE79          SI10
AE79 A5 C5    LDA    DUEFLQ
AE7B 10 A3    BPL    SBAL15
; BAL = BALLOON PAYMENT
; ENTER VALUE

AE7D 20 B0 A3 ;JSR    LDPV
AE80 20 B6 DD ;JSR    FMOVE
AE83 20 9A A3 ;JSR    LDFV
AE86 20 3A A9 ;JSR    SFDIV
AE89 20 BB 9F ;JSR    FPUSHO
AE8C 20 AB A3 ;JSR    LDN
AE8F 20 24 A9 ;JSR    SRECIP
; FV/PV
; 1/N

```



```

AE89 20 B8 A6 JSR FPUSH0
AE8C 20 A8 A3 JSR LDN
AE8F 20 24 A9 JSR SRECIP , 1/N

```

COLLEEN CALCULATOR, BY C SHAW

```

AE92 20 19 A6 JSR SPOWER
AE95 20 8D A6 JSR SUBONE
AE98 4C 63 AE JMP SIO5 , STORE NEW I
, IF ASMBL , ANNUITY - USE NEWTON - RAPHSON ITERATION (SEE SSGRT)

SI20 CMP #1
BEQ SI30
JMP KEYERR

SI30 , ORDINARY ANNUITY/FV
, F(I) = PMT*((1+I)^N-1)/I - FV = 0
, FPRIME(I) = (PMT*N*(1+I)^(N-1)-(F(I)+FV))/I
, DELTA I = F(I)/FPRIME(I)
LDA #0 , I = MEM(6) <- .01 = $3F,1,0,0,0,0
LDX #3

SILP1 STA 6*FPREC+MEMORY+2, X
DEX
BPL SILP1
LDA #$3F
STA 6*FPREC+MEMORY
LDA #1
STA 6*FPREC+MEMORY+1
STA DMFLG , DON'T DISPLAY MEM DURING ITERATION (<-1)
LDA #$FF , NUMBER OF ITERATIONS
STA ITER

SILP JSR Z1IN1I , ((1+I)^N-1)/I
LDA #8 , PMT
JSR MEMMUL
JSR FPUSH0 , SAVE FOR SPSUB
JSR FSTOT , SAVE FOR F'(I)
JSR LDFV
JSR SPSUB
JSR FPUSH0 , SAVE F(I)
JSR Z1IN1 , (1+I)^(N-1)
LDA #8 , PMT
JSR MEMMUL
LDA #7 , N
JSR MEMMUL
JSR FLD1T , RELOAD F(I)-PV
JSR SFSUB
JSR FPUSH0
JSR LDI
JSR SPDIV , F'(I)
JSR SPDIV , F(I)/F'(I) = DELTA I
LDA FRO , 0?
BNE SI35 , NO. CONTINUE
JSR LDI , YES. RELOAD I INTO X REG
JMP SI40 , DONE

SI35 JSR FMOVE , NO. I<- I+DELTA I
JSR LDI
JSR SFSUB
DEC ITER
BEQ SI40 , DONE
JSR SIO5 , STORE NEW I
JMP SILP , CONTINUE
SI40 DEC DMFLG , <- 0 DONE => RETURN
JMP SIO5 , STORE NEW I & DISPLAY
ENDIF

```

```

AE9B          SN
AE9B A5 C6    LDA    ENTFLQ
AE9D F0 46    BEQ    SN05
AE9F A5 C5    LDA    DUEFLQ
AEA1 10 1B    BPL    SN20
AEA3 20 80 A3 JSR    LDPV
AEA6 20 B6 DD JSR    FMOVE
AEA9 20 9A A3 JSR    LDFV
AEAC 20 3A A9 JSR    SPDIV
AEAF 20 B3 A4 JSR    SLN
AEB2 20 BB 9F JSR    FPUSHO
AEB5 20 B0 A6 JSR    ZLN1I
AEBB 20 37 A9 JSR    SPDIV
AEBB 4C E5 AE JMP    SN05
AEBE          SN20
AEBE 29 02    AND    #2
AEC0 D0 2B    BNE    SN50
;
AEC2 20 9A A3 JSR    LDFV
AEC5 A9 06    LDA    #6
AEC7 20 E6 A3 JSR    MEMMUL
AEC9 20 BB 9F JSR    FPUSHO
AECD 20 AC A3 JSR    LDPMT
AED0 20 B9 AB JSR    ZMUL1I
AED3 20 37 A9 JSR    SPDIV
AED6 20 51 A9 JSR    ONEADD
AED9          SN30
AED9 20 B3 A6 JSR    SLN
AEDC 20 BB 9F JSR    FPUSHO
AEDF 20 B0 A6 JSR    ZLN1I
AEE2 20 37 A9 JSR    SPDIV
AEE5 A9 07    SN05   LDA    #7
AEE7 4C 4A AE JMP    MEMSTO
;
AEEA          SN50
;
AEEA 20 AC A3 JSR    LDPMT
AEEA 20 B9 AB JSR    ZMUL1I
AEED 20 55 9F JSR    FSTOT
AEF0 20 BB 9F JSR    FPUSHO
AEF6 20 9E A3 JSR    LDI
AEF9 20 B6 DD JSR    FMOVE
AEFC A9 04    LDA    #4
AEFE 20 E6 A3 JSR    MEMMUL
AF01 20 80 A9 JSR    SPSUB
AF04 20 BB 9F JSR    FPUSHO
AF07 20 9E A3 JSR    LDI
AF0A A9 09    LDA    #9
AF0C 20 E6 A3 JSR    MEMMUL
AF0F 20 B6 DD JSR    FMOVE
AF12 20 49 9F JSR    FLDOT
AF15 20 83 A9 JSR    SFSUB
AF18 20 37 A9 JSR    SPDIV
AF1B 4C D9 AE JMP    SN30
;
AF1E          SPMT
; PMT = PAYMENT

```

; N = NUMBER OF PERIODS
 ; COMPOUND INTEREST $N = \text{LN}(FV/PV) / \text{LN}(1+I)$
 ; FV/PV
 ; $\text{LN}(1+I)$
 ; STORE NEW N
 ; ANNUITY FV OR PV?
 $1 \Rightarrow PV$
 $0 \Rightarrow FV \quad N = \text{LN}(FV * I / \text{PMT} + 1) / \text{LN}(1+I)$
 DUE $N = \text{LN}(FV * I / (\text{PMT} * (1+I) + 1) / \text{LN}(1+I)$
 ; I
 ; IF DUE THEN $*(1+I)$
 ; $\text{LN}(1+I)$
 ; ENTER
 $PV \cdot N = \text{LN}((\text{PMT} - I * \text{BAL}) / (\text{PMT} - I * \text{PV})) / \text{LN}(1+I)$
 $\text{DUE} \cdot N = \text{LN}((\text{PMT} * (1+I) - I * \text{BAL}) / (\text{PMT} * (1+I) - I * \text{PV})) / \text{LN}(1+I)$
 ; IF DUE THEN $*(1+I)$
 ; BAL
 ; PV
 ; RELOAD PMT OR $\text{PMT} * (1+I)$

AF1B 20 07
AF1B 4C D9 AE

JMP

SN30

AF1E

SPMT

;PMT = PAYMENT

COLLEEN CALCULATOR, BY C SHAW

```
AF1E A5 C6      LDA      ENTFLG
AF20 F0 3F      BEQ      SPMT05
AF22 A5 C5      LDA      DUEFLG
AF24 10 03      BPL      SPMT20
AF26 4C 20 AE   JMP      SBAL15      ;NO PMT IF COMPOUNT INTEREST - NOT VALID COMMAND
AF29            SPMT20
AF29 29 02      AND      #2
AF2B D0 17      BNE      SPMT30

;
;
AF2D 20 E1 AD   JSR      Z11NM1
AF30 20 89 A8   JSR      ZMUL1I
AF33 20 B6 DD   JSR      FMOVE
AF36 20 9A A3   JSR      LDFV
AF39 20 3A A9   JSR      SFDIV
AF3C A9 06      LDA      #6          ; I
AF3E 20 E6 A3   JSR      MEMMUL
AF41 4C 61 AF   JMP      SPMT05
AF44            SPMT30
;
AF44 20 21 A9   JSR      Z11MN
AF47 A9 04      LDA      #4
AF49 20 E6 A3   JSR      MEMMUL
AF4C 20 B6 DD   JSR      FMOVE
AF4F 20 B0 A3   JSR      LDPV
AF52 20 83 A9   JSR      SFSUB
AF55 20 BB 9F   JSR      FPUSHO
AF58 20 E7 AD   JSR      Z11INI
AF5B 20 89 A8   JSR      ZMUL1I
AF5E 20 37 A9   JSR      SPDIV
AF61 A9 08      LDA      #8
AF63 4C 4A AE   JMP      MEMSTO      ;ENTER

;
;
AF66            SPV
AF66 A5 C6      LDA      ENTFLG
AF68 F0 28      BEQ      SPV05
AF6A A5 C5      LDA      DUEFLG
AF6C 10 0B      BPL      SPV20
;
AF6E 20 21 A9   JSR      Z11MN
AF71 A9 05      LDA      #5
AF73 20 E6 A3   JSR      MEMMUL
AF76 4C 92 AF   JMP      SPV05
AF79            SPV20
;
AF79 20 E7 AD   JSR      Z11INI
AF7C A9 08      LDA      #8
AF7E 20 E6 A3   JSR      MEMMUL
AF81 20 89 A8   JSR      ZMUL1I
AF84 20 BB 9F   JSR      FPUSHO
AF87 20 21 A9   JSR      Z11MN
AF8A A9 04      LDA      #4
AF8C 20 E6 A3   JSR      MEMMUL
AF8F 20 67 A9   JSR      SPADD
AF92 A9 09      LDA      #9
AF94 4C 4A AE   JMP      MEMSTO      ;ENTER

;PV = PRESENT VALUE
;PV = PMT * (1-(1+I)^-N)/I+BAL*(1+I)^-N
;FV
;STORE NEW VALUE
;PV = PMT * (1-(1+I)^-N)/I+BAL*(1+I)^-N
;DUE = PMT * (1-(1+I)^-N)/I*(1+1)+BAL*(1+I)^-N
;(1-(1+I)^-N)/I
;PMT
;IF DUE THEN *(1+I)
;(1+I)^-N
;BAL
;ENTER
```

STATISTICS ROUTINES

```

;
;
; MEM(A) <- MEM(A) + TOS      MEM(A+1) <- MEM(A+1) + SQU(TOS)
AF97      ZSIGMA
AF97 48      PHA
AF98 20 18 9F JSR      FLDOS      ; FRO <- TOP OF STACK(TOS)
AF98 68      PLA
AF9C 48      PHA
AF9D 20 AA AF JSR      MEMADD
AFA0 20 18 9F JSR      FLDOS
AFA3 20 5A AB JSR      SSQUAR
AFA6 68      PLA
AFA7 18      CLC
AFA8 69 01      ADC      #1      MEM(A) <- MEM(A) + /- FRO

; MEMADD
AFAA      MEMADD
AFAA 20 88 A3 JSR      MEMLD1
AFAD A5 C8      LDA      MEMFLG      ; ADD?
AFAF D0 06      BNE      MEMA10      ; NO
AFB1 20 6A A9 JSR      SFADD      ; YES
AFB4 4C BD AF JMP      MEMA20
AFB7      MEMA10
AFB7 20 83 A9 JSR      SFSUB      ; SUBTRACT
AFBA 20 82 A5 JSR      SCHGSG      ; FRO - MEM(A) -> MEM(A) - FRO
AFBD      MEMA20
AFBD 20 C0 A3 JSR      MEMLDR
AFCD 20 A7 DD JSR      FSTOR      ; MEM(MEMNUM) <- FRO
AFC3 A5 BD      LDA      DSPFLG
AFC5 D0 06      BNE      MEMA30
AFC7 20 86 9D JSR      TOKNUM      ; DISPLAY IN MEM AREA
AFCA 4C 72 AB JMP      DSPMEM
AFCD      MEMA30
AFCD 60      RTS

; MEMDIV
AFCE      MEMDIV
AFCE 20 B2 A3 JSR      MEMLD0      ; FRO <- MEM(A) / N
AFD1 A9 04      LDA      #4      ; MEM(A) -> FRO
AFD3 20 88 A3 JSR      MEMLD1      ; N -> FR1
AFD6 4C 3A A9 JMP      SFDIV

```


COLLEEN CALCULATOR BY C SHAW

```

AFD9 A9 05      SXMEAN LDA #5      ; MEAN(X) <- SIGMA(X)/N
AFD1 D0 F1      BNE MEMDIV ; JMP
AFDD A9 07      SYMEAN LDA #7      ; MEAN(Y) <- SIGMA(Y)/N
AFDF D0 ED      BNE MEMDIV ; JMP
AFE1            SXVARI          ; VARIANCE(X) <- (SIGMA(SQU(X))-SQU(SIGMA(X)))/N/(N+WEIGHT)
AFE1 A9 05      LDA #5      ; SIGMA X
AFE3 D0 02      BNE ZVAR     ; JMP
AFE3            XYVARI          ; VAR(Y) <- (SIGMA(SQU(Y))-SQU(SIGMA(Y)))/N/(N+WEIGHT)
AFE5 A9 07      LDA #7      ; SIGMA Y
AFE7            ZVAR           ; THIS PART IS COMMON TO BOTH SXVARI AND XYVARI
AFE7 20 C9 9F   JSR ZVAR2    ; COMPUTE SIGMA(SQU())-SQU(SIGMA())/N
AFE7 20 03      LDA #3      ; WEIGHT
AFEC 20 B8 A3   JSR MEMLD1
AFEF 20 6A A9   JSR SFADD    ; N+WEIGHT (SHOULD BE 0 OR -1)
AFF2 4C 37 A9   JMP SPDIV    ; NUMERATOR/(N+WEIGHT)

AFF5 20 EC BF   SCORRE JSR SXSTDD ; CORRELATION = R = M * STDDEV(X)/STDDEV(Y)
AFF8 20 B8 9F   JSR FPUSHO
AFFB 20 F2 BF   JSR SYSTDD
AFFE 20 37 A9   JSR SPDIV    ; STDDEV(X)/STDDEV(Y)
B001 20 B8 9F   JSR FPUSHO
B004 20 3A B0   JSR SSLOPE   ; M
B007 4C 94 A8   JMP SPMUL

B00A            SX            ; X <- (Y-B)/M (Y ENTERED IN X REG)
B00A 20 B8 9F   JSR FPUSHO   ; SAVE Y
B00D 20 1C B0   JSR SYINTE   ; B
B010 20 B0 A9   JSR SPSUB    ; Y-B
B013 20 B8 9F   JSR FPUSHO   ; M
B016 20 3A B0   JSR SSLOPE   ; (Y-B)/M
B019 4C 37 A9   JMP SPDIV

B01C            SYINTE        ; Y-INTERCEPT = B = (SIGMA(Y)-M*SIGMA(X))/N
B01C 20 3A B0   JSR SSLOPE   ; M
B01F A9 05      LDA #5      ; SIGMA X
B021 20 B8 A3   JSR MEMLD1   ; M*SIGMA(X)
B024 20 97 A8   JSR SFMUL
B027 20 B6 DD   JSR FMOVE
B02A A9 07      LDA #7      ; SIGMA(Y)
B02C 20 B2 A3   JSR MEMLD0
B02F 20 B3 A9   JSR SFSUB
B032 A9 04      LDA #4      ; N
B034 20 B8 A3   JSR MEMLD1
B037 4C 3A A9   JMP SFDIV

B03A            SSLOPE        ; SLOPE = M = (SIGMA(X*Y)-SIGMA(X)*SIGMA(Y)/N)/(SIGMA(SQU(X))-SQU(SIGMA(X)))/N
B03A A9 05      LDA #5      ; SIGMA(X)
B03C 20 C9 9F   JSR ZVAR2    ; COMPUTE SIGMA(SQU(X))-SQU(SIGMA(X))/N, STORE ON STACK
B03F 20 B6 DD   JSR FMOVE    ; N-> FR1 (PUT IN FRO BY ZVAR2)
B042 A9 05      LDA #5      ; SIGMA(X)
B044 20 B2 A3   JSR MEMLD0
B047 20 3A A9   JSR SFDIV
B04A A9 07      LDA #7      ; SIGMA(Y)
B04C 20 B8 A3   JSR MEMLD1
B04F 20 97 A8   JSR SFMUL
B052 20 B6 DD   JSR FMOVE
B055 A9 09      LDA #9      ; SIGMA(X*Y)
B057 20 B2 A3   JSR MEMLD0

```

COLLEEN CALCULATOR, BY C SHAW

B05A 20 83 A9
B05D 20 86 9F
B060 4C 3A A9

JSR SFSUB
JSR FPOP1
JMP SFDIV

; NUMERATOR
; LOAD DENOMINATOR (FROM ZVAR2)

B063
B063 A9 03
B065 85 A3
B067 4C 65 AB

SNWEIG

LDA #3
STA MEMNUM
JMP SST010

; WEIGHT FACTOR

; MEM(3) ← X

COLLEEN CALCULATOR, BY C SHAW

```

; BASIC SINE ROUTINE
; TO FIX BUGS OF VERSION 5.9 OF SHEP BASIC
;
; BY DAVE & LARRY
; 4-6-79
;
;
; SINE ROUTINE
; COMPUTE QUADRANT, GET FRACTION AND DO POLYNOMIAL,
; THEN ADJUST FOR QUADRANT
SSIN
B06A          JSR     SINMOD      ;TAKE ANGLE MOD 2*PI, 360 OR 400
B06A 20 DB BF LDA     FRO        GET SIGN
B06D A5 D4     AND     ##80
B06F 29 80     STA     FCHFLG     AND SAVE
B071 85 F0     LDA     FRO
B073 A5 D4     AND     ##7F      FRO=ABS(FRO)
B075 29 7F     STA     FRO
B077 85 D4

; FRO=FRO/(PI/2) OR FRO=FRO/90
B079 20 F1 A3 JSR     PIOVL      ;LOAD X & Y REGS TO GET PI/2 90 OR 100
B07C 20 B6 AD JSR     LD1DIV     FRO=FRO/FR1
B07F 90 04     BCC     NOSNER
B081 60        RTS              ;RETURN
B082          SINERR
B082 4C 88 A9 JMP     CRYSDN      GO IF ERROR
B085          NOSNER

; IF FRO NOW FRACTION, IT IS QUADRANT 0
; ELSE, GET INTEGER OF FRO LSD
B085 A9 00     LDA     #0
B087 85 B8     STA     QUADFLG     ASSUME QUADRANT 0
B089 38        SEC
B08A A5 D4     LDA     FRO        GET EXPONENT
B08C E9 40     SBC     ##40       SUBTRACT 64 EXCESS
B08E 30 37     BMI     SINF3      GO IF QUADRANT 0
B090 C9 04     CMP     #FPREC-2   IS EXPONENT TOO BIG?
B092 B0 EE     BCS     SINERR     YES

; ACC=INDEX TO LSD. GET 10*TEN'S DIGIT + ONE'S DIGIT
; THEN AND WITH 3 TO GET QUADRANT
B094 AA        TAX
B095 B5 D5     LDA     FRO+1,X     GET LSD
B097 29 0F     AND     ##F        GET ONE'S DIGIT
B099 85 F1     STA     DIGRT      AND SAVE
B09B B5 D5     LDA     FRO+1,X     GET LSD
B09D 29 F0     AND     ##F0       GET TEN'S DIGIT
B09F 4A        LSR     A          TIMES 8
B0A0 85 B8     STA     QUADFLG     AND TEMP SAVE
B0A2 4A        LSR     A          TIMES 2
B0A3 4A        LSR     A
B0A4 18        CLC
B0A5 65 B8     ADC     QUADFLG     PLUS TIMES 8 GIVES TIMES 10
B0A7 65 F1     ADC     DIGRT      PLUS ONE'S DIGIT GIVES INTEGER
B0A9 29 03     AND     #3         MASK LOW BITS
B0AB 85 B8     STA     QUADFLG     NOW HAVE QUADRANT (0,1,2, OR 3)
B0AD 86 F1     STX     DIGRT      SAVE INDEX TO LSD

```

```

; PUT FRO IN FR1, AND CLEAR FRACTIONAL PART OF FR1
; THEN GET FRO=FRACTIONAL PART OF FRO
BOAF 20 B6 DD      JSR      FMOVE      FR1=FRO
BOB2 A6 F1         LDX      DIGRT      RESTORE INDEX
BOB4 A9 00         LDA      #0
BOB6 95 E2         SINFI  STA      FR1+2,X  CLEAR FRACTIONAL PART
BOB8 E8           INX                      FROM DIGRT+1 TO END
BOB9 E0 04         CPX      #FPREC-2    DONE?
BOBB 90 F9         BCC      SINFI       NO
BOBD 20 60 DA      JSR      FSUB        FRO=FRO-FR1 (FRO WILL BE FRACTIONAL PART)

; IF ODD QUADRANT, SET FRO=1-FRO (90 DEGREE INVERT)
BOC0 46 B8         LSR      QUADFLG     IS IT ODD QUADRANT?
BOC2 90 03         BCC      SINF3       NO
BOC4 20 73 A9      JSR      ONESUB      ; FRO <- 1-FRO

; SAVE ARG FOR LATER
BOC7              SINFI 3
BOC7 A2 E6         LDX      #FPSCR      ; CAN'T USE FTEMP BECAUSE SSIN IS CALLED BY SPOLAR
BOC9 A0 05         LDY      #FPSCR/256
BOCB 20 A7 DD      JSR      FSTOR      ; FPSCR <- FRO

; NOW COMPUTE SINE
; THIS CODE TAKEN FROM BASIC 5.9 LINES 6760-6770
BOCE 20 5A AB      JSR      SSQUAR      FRO=X**2
BOD1 B0 31         BCS      SINFIN      ; ERROR (ALREADY REPORTED)
BOD3 A9 06         LDA      #NSCF
BOD5 A2 06         LDX      #SCDEF
BOD7 A0 BA         LDY      #SCDEF/256
BOD9 20 40 DD      JSR      PLYEVL      EVALUATE P(X**2)
BODC A2 E6         LDX      #FPSCR
BODE A0 05         LDY      #FPSCR/256
BOE0 20 C5 AD      JSR      LD1MUL      FRO=SIN(X)=X*P(X**2)

; IF LOWER QUADRANT (2 OR 3) THEN FRO=-(FRO)
BOE3 46 B8         LSR      QUADFLG     IS IT LOWER QUAD?
BOE5 90 08         BCC      SINF4       NO
BOE7 A5 D4         LDA      FRO         IS FRO=0
BOE9 F0 0C         BEQ      SINDON      YES
BOEB 49 80         EOR      #$80        ELSE, FRO=-(FRO)
BOED 85 D4         STA      FRO

; IF SIGN WAS NEGATIVE COMING IN TO ROUTINE, INVERT SIGN
; GOING OUT
BOEF A5 D4         SINFI 4 LDA      FRO      ANSWER
BOF1 F0 04         BEQ      SINDON      GO IF ZERO
BOF3 45 F0         EOR      FCHRFLG     INVERT ORIGINAL SIGN
BOF5 85 D4         STA      FRO        AND THIS IS END ANSWER

; IF ABS(FRO) >= 1 THEN PERFORM PSEUDO INT(FRO)
BOF7 29 7F         SINDON AND      #$7F      WITHOUT SIGN BIT
BOF9 C9 40         CMP      #$40        COMPARE $40
BOFB 90 07         BCC      SINFIN
BOFD 1B           CLC                      NO ERROR CLEAR
BOFE A9 00         LDA      #0          STORE 0 IN LOW BYTES OF FRO
B100 85 D8         STA      FRO+4
B102 85 D9         STA      FRO+5
B104 60           SINFI  RTS

```


COLLEEN CALCULATOR, BY C SHAW

Address	Hex	Label	Op	Op2	Comment
B105	20 8C 7F	SASIN	JSR	ARCSUB	; ARCSIN(FRO) = ARCTAN(FRO/SQRT(FRO*FRO))
B108	A5 E0		LDA	FR1	; FR1 <- SQRT(1-FRO*FRO)
B10A	D0 0C		BNE	SAS10	
B10C	A5 D4		LDA	FRO	; !FRO! = 1
B10E	08		PHP		
B10F	20 A0 A4		JSR	SAC10	; FRO <- 90 OR PI/2
B112	28		PLP		
B113	10 EF		BPL	SINFIN	; RETURN FRO = +1. ARCSIN(+1) = 90 OR PI/2
B115	4C 82 A5		JMP	SCHGSG	; FRO = -1. ARCSIN(-1) = -90 OR -PI/2
B118		SAS10			
B118	20 3A A9		JSR	SFDIV	
			JMP	SATAN	
					FROM SHEPARDSON ATARI BASIC 5.9 4-5-79 (MODIFIED)
B11B		SATAN			; ARCTAN(FRO)
B11B	A9 00		LDA	#0	
B11D	85 F0		STA	FCHRFLG	; SIGN FLAG OFF
B11F	85 F1		STA	DIORT	; AND TRANSFORM FLAG
B121	A5 D4		LDA	FRO	
B123	29 7F		AND	##7F	
B125	C9 40		CMP	##40	; CHECK X VS 1.0
B127	30 15		BMI	ATAN1	; X<1 - USE SERIES DIRECTLY
B129	A5 D4		LDA	FRO	; X>=1 - SAVE SIGN & TRANSFORM
B12B	29 80		AND	##80	
B12D	85 F0		STA	FCHRFLG	; REMEMBER SIGN
B12F	E6 F1		INC	DIORT	
B131	A9 7F		LDA	##7F	
B133	25 D4		AND	FRO	
B135	85 D4		STA	FRO	; FORCE PLUS
B137	A2 EA		LDX	##FP9S	
B139	A0 DF		LDY	##FP9S/\$100	
B13B	20 95 DE		JSR	XFORM	; CHANGE ARG TO (X-1)/(X+1)
B13E		ATAN1			
					ARCTAN(X), -1<X<1 BY SERIES APPROX
B13E	A2 E6		LDX	##FPSR	; CAN'T USE FTEMP BECAUSE SATAN IS CALLED BY OTHER ROUTINES WHICH USE IT
B140	A0 05		LDY	##FPSR/256	
B142	20 A7 DD		JSR	FSTOR	; X->FTEMP (CALC, NOT SHEP REG)
B145	20 5A AB		JSR	SSQUAR	; X*X -> FRO
B148	B0 BA		BCS	SINFIN	; OVERFLOW (ERROR ALREADY REPORTED)
B14A	A9 0B		LDA	##NATCF	
B14C	A2 AE		LDX	##ATCOEF	
B14E	A0 DF		LDY	##ATCOEF/256	
B150	20 40 DD		JSR	PLYEVL	; P(X*X)
B153	B0 26		BCS	ATNOUT	; ERROR
B155	A2 E6		LDX	##FPSR	
B157	A0 05		LDY	##FPSR/256	
B159	20 C5 AD		JSR	LD1MUL	; X*P(X*X)
B15C	B0 A6		BCS	SINFIN	; OVERFLOW (ERROR ALREADY REPORTED SO RETURN)
B15E	A5 F1		LDA	DIORT	; WAS ARG XFORMED
B160	F0 10		BEQ	ATAN2	; NO.
B162	A2 F0		LDX	##PIOV4	; YES-ADD ARCTAN(1) = PI/4
B164	A0 DF		LDY	##PIOV4/256	
B166	20 98 DD		JSR	FLD1R	
B169	20 66 DA		JSR	FADD	
B16C	A5 F0		LDA	FCHRFLG	; GET ORG SIGN
B16E	05 D4		DRA	FRO	
B170	85 D4		STA	FRO	; ATAN(-X) = -ATAN(X)
B172		ATAN2			
B172	A5 FB		LDA	RADFLG	; RAD OR DEG
B174	F0 05		BEQ	ATNOUT	; RAD - FINI

		CLC	
		ADC	#PIOV1B-FPREC
		TAX	
B176 A2 36		LDX	
B178 4C B4 AD		JMP	
B17B	ATNOUT		
B178 4C 86 A9		JMP	CRYCHK

; DIVIDE BY PI/180 OR PI/200
; ABOVE IS USED IF GRADS ALLOWED
; FRO <- FRO/(WHATEVER)


```

B17E          SPOLAR  ; ->RECT Y=THETA X=R  NEW Y=R*SIN(THETA)  NEW X=SQRT(SQU(R)-SQU(Y))
B17E A9 7D          LDA      #ZPOLAR
B180 20 F0 9B      JSR      PUTMSG          ; DISPLAY "->RECTANGULAR"
B183 20 55 9F      JSR      FSTOT          ; SAVE X=R
B186 20 9D 9F      JSR      FPOPO          ; LOAD THETA=Y
B189 20 6A B0      JSR      SSIN           ; SIN(THETA)
B18C 20 4F 9F      JSR      FLDIT          ; RELOAD R
B18F 20 97 AB      JSR      SFMUL          ; Y = R*SIN(THETA)
B192 20 BB 9F      JSR      FPUSHO         ; SAVE NEW Y ON STACK
B195 20 5A AB      JSR      SSQUAR         ; Y*Y
B198 20 BB 9F      JSR      FPUSHO         ; SAVE Y*Y
B19B 20 49 9F      JSR      FLDOT          ; LOAD R
B19E 20 5A AB      JSR      SSQUAR         ; R*R
B1A1 20 86 9F      JSR      FPOP1          ; RELOAD Y*Y
B1A4 20 83 A9      JSR      SFSUB          ; R*R - Y*Y

```

FROM SHEPARDSON ATARI BASIC 5.9 4-5-79 (MODIFIED)

USES NEWTON-RAPHSON ITERATION

$F(Y) = Y*Y - X$

$FPRIME(Y) = 2*Y$

$Y[I+1] = Y[I] - F(Y[I]) / FPRIME(Y[I]) = Y[I] + .5*((X/Y[I]) - Y[I])$

$XC = SQRT(X)$

```

B1A7          SSQRT  LDA      #0
B1A7 A9 00          STA      DIGRT
B1A9 85 F1          LDA      FRO
B1AB A5 D4          BPL      SQRO
B1AD 10 06          JSR      BITERR          ; <0 => ERROR
B1AF 20 95 A3      JSR      SABSVA          ; TAKE ABS VALUE AND DO SQUARE ROOT (ABSVAL LOADS FRO INTO A)
B1B2 20 49 B2
B1B5          SGRO  CMP      #3F
B1B5 C9 3F          BEQ      FSQR          ; X IN RANGE OF APPROX - GO DO IT TO IT
B1B7 F0 17          CLC
B1B9 18            ADC      #1
B1BA 69 01          STA      DIGRT          ; NOT IN RANGE - TRANSFORM
B1BC 85 F1          STA      FR1          ; MANTISSA = 1
B1BE 85 E0          LDA      #1
B1C0 A9 01          STA      FR1+1
B1C2 85 E1          LDX      #FPREC-3          ; CHANGED 5/11/79 FROM FPREC-2
B1C4 A2 03          LDA      #0
B1C6 A9 00
B1C8          SQR1  STA      FR1+2, X
B1C8 95 E2          DEX
B1CA CA            BPL      SQR1
B1CB 10 FB          JSR      FDIV          ; X/100**N
B1CD 20 28 D3      ; SQR(X) 0.1<=X<1
B1D0          FSQR  LDA      #6
B1D0 A9 06          STA      ESIGN
B1D2 85 EF          LDX      #FSCR
B1D4 A2 E6          LDY      #FSCR/256
B1D6 A0 05          JSR      FSTOR          ; STASH X IN FSCR
B1D8 20 A7 DD      LDA      #2
B1DB A9 02          JSR      INTSUB          ; 2-X
B1DD 20 75 A9      LDX      #FSCR
B1E0 A2 E6          LDY      #FSCR/256
B1E2 A0 05          JSR      LD1MUL          ; X*(2-X) : 1ST APPROX
B1E4 20 C5 AD
B1E7          SGRLP LDX      #FSCR1          ; DON'T USE FTEMP BECAUSE SSQRT IS USED BY OTHER ROUTINES
B1E7          LDY      #FSCR1/256
B1E9 A0 05          JSR      FSTOR          ; Y->FSCR1
B1EB 20 A7 DD

```

B1EE 20 B6 DD	JSR	FMOVE	; Y->FR1
B1F1 A2 E6	LDX	#FSCR	
B1F3 A0 05	LDY	#FSCR/256	
B1F5 20 89 DD	JSR	FLDOR	
B1FB 20 28 DB	JSR	FDIV	; X/Y
B1FB A2 EC	LDX	#FSCR1	
B1FD A0 05	LDY	#FSCR1/256	
B1FF 20 98 DD	JSR	FLD1R	
B202 20 60 DA	JSR	FSUB	; (X/Y)-Y
B205 A2 6C	LDX	#FHALF	
B207 A0 DF	LDY	#FHALF/256	
B209 20 C5 AD	JSR	LD1MUL	; 5*((X/Y)-Y)=DELTA Y
B20C A5 D4	LDA	FRO	; DELTA 0
B20E F0 0E	BEG	SQRDON	
B210 A2 EC	LDX	#FSCR1	
B212 A0 05	LDY	#FSCR1/256	
B214 20 98 DD	JSR	FLD1R	
B217 20 66 DA	JSR	FADD	; Y=Y+DELTA Y
B21A C6 EF	DEC	ESIGN	; COUNT & LOOP
B21C 10 C9	BPL	SQRLP	
B21E	SQRDON		
B21E A2 EC	LDX	#FSCR1	; DELTA = 0 - GET Y BACK
B220 A0 05	LDY	#FSCR1/256	
B222 20 89 DD	JSR	FLDOR	
			WAS ARG TRANSFORMED?
B225 A5 F1	LDA	DIGRT	
B227 F0 20	BEG	SQROUT	; NO FINI
B229 38	SEC		
B22A E9 40	SBC	##40	
			; YES - TRANSFORM RESULT TO MATCH
B22C 4A	LSR	A	; DIVIDE EXP BY 2
B22D 18	CLC		
B22E 69 40	ADC	##40	
B230 85 E0	STA	FR1	
B232 A5 F1	LDA	DIGRT	
B234 6A	ROR	A	
B235 A9 01	LDA	#1	; MANTISSA = 1
B237 90 02	BCC	SQR2	; WAS EXP_ODD OR EVEN
B239 A9 10	LDA	##10	; ODD - MANT = 10
B23B	SQR2		
B23B 85 E1	STA	FR1+1	
B23D A2 03	LDX	#FPREC-3	; CHANGED 5/11/79 FROM FPREC-2
B23F A9 00	LDA	#0	
B241	SQR3		
B241 95 E2	STA	FR1+2, X	; CLEAR REST OF MANTISSA
B243 CA	DEX		
B244 10 FB	BPL	SQR3	
B246 20 97 AB	JSR	SFMUL	; SQR(X) = SQR(X/100**N) * (10**N)
B249	SQROUT		
B249	SABSV A		; FRO <- ABSVAL(FRO) A<-FRO
B249 A5 D4	LDA	FRO	
B24B 29 7F	AND	##7F	
B24D 85 D4	STA	FRO	
B24F 60	RTS		


```

      .IF      ASMBL      ; THE FOLLOWING HAVE BEEN REMOVED:
      ;
      ; INPUT: FRO IN MMDD.YYYY FORMAT
      ; Z = YYYY; IF (MM-1) <=1 THEN Z=Z-1;
      ; OUTPUT: FRO = FACTOR = 365*YYYY + DD + 31*(MM-1) - (DAYTRM, (MM-1))
      ;                               +INT(Z/4) - INT(.75*INT(Z/100)+11)
      DAYERR
      LDA      #DAYMSG
      JSR      ERRSUB
      JSR      PCLRO      ; CLEAR X
      SEC
      RTS      ; INDICATE ERROR

DAYSUB
      LDA      FRO
      BMI      DAYERR      ; MUST BE >0
      JSR      FSTOT
      JSR      SINTEG
      ;
      LDA      #100
      JSR      INTMOD
      ;
      LDA      MODFAC+1
      BEQ      DAYERR      ; CHECK MM AND DD
      ; MM = 0 => ERROR
      CMP      #$13
      BCS      DAYERR      ; MM > 12 => ERROR
      SED
      ; MM <- MM-1 (0-$11)
      SBC      #1-1
      STA      MODFAC+1
      CLD
      CMP      #$10
      BCC      DAYS10      ; DEC -> INTEGER
      SBC      #6
      DAYS10
      TAX
      LDA      FRO+1
      BEQ      DAYERR      ; DD = 0 => ERROR
      CMP      MAXDAY, X
      BCS      DAYERR      ; DD TOO LARGE
      LDA      DAYTRM, X
      STA      DAYTMP      ; SAVE INT(.4MM+2.3)
      JSR      FPUSHO
      JSR      FLDOT
      JSR      SFRACT
      LDX      #C10000
      LDY      #C10000/256
      JSR      FLD1R
      JSR      SFMUL
      JSR      SINTEG
      LDA      FRO
      CMP      #$41
      BNE      DAYERR      ; MUST HAVE $41, YY, YY, 0, 0, 0, 0
      LDA      FRO+1
      CMP      #$16
      BCC      DAYERR      ; YYYY MUST BE GE 1600
      JSR      FSTOT
      ; FTEMP <- YYYY
      LDX      #C365
      ; 365 * YYYY

```

```

LDY      #C365/256
JSR      FLD1R
JSR      SFMUL

JSR      FPOP1          ; DD
JSR      SFADD          ; 365*YYYY + DD

LDX      #MODFAC
LDY      #MODFAC/256    ; LOAD MM-1
JSR      FLD1R
JSR      FPUSH0
LDA      #31
JSR      PSET0
JSR      SFMUL          ; 31*(MM-1)
JSR      FPOP1
JSR      SFADD          ; 365*YYYY + DD + 31*(MM-1)

LDA      MODFAC+1
CMP      #2             ; JAN OR FEB?
BCS      DAYS20         ; NO.
LDA      FTEMP+2        ; YES. YYYY <- YYYY-1
SED
SBC      #1-1           ; CARRY IS CLEAR
STA      FTEMP+2
BCS      DAYS15
LDA      FTEMP+1
SBC      #1-1
STA      FTEMP+1

DAYS15   CLD
DAYS20   JSR      FMOVE          ; ADD -(DAYTRM, (MM-1))
LDA      DAYTMP
JSR      PSET0
JSR      SCHGSG
JSR      SFADD

JSR      FPUSH0          ; ADD INT(YYYY/4)
LDA      #4
JSR      PSET0
JSR      FMOVE
JSR      FLD0T
JSR      SFDIV
JSR      SINTEG
JSR      FPOP1
JSR      SFADD

JSR      FPUSH0          ; SUB INT(.75[INT[(YYYY/100)+1]])
LDA      #100
JSR      PSET0
JSR      FMOVE
JSR      FLD0T          ; YYYY
JSR      SFDIV          ; YYYY/100
LDA      #1
JSR      INTADD          ; 1+YYYY/100
JSR      SINTEG
LDX      #CPT75
LDY      #CPT75/256
JSR      FLD1R
JSR      SFMUL

```


JSR FLD1R
JSR SFMUL

COLLEEN CALCULATOR, BY C SHAW

JSR SINTEG
JSR FMOVE
JSR FPOPO
JMP SFSUB
.PAGE

HYP SUB

; FRO <- EXPE(X), FR1 <- EXPE(-X)
FOR COSH, SINH

JSR FPUSHO
JSR SCHGSG
JSR SEXPE
JSR SXCHGY
JSR SEXPE
JMP FPOP1

; EXPE(-X)

; EXPE(X)

SCOMBI

; Y COMBI X = C(N,R) = N!/(R!(N-R)!) = P(N,R)/R! = (Y PERMU X)/X!

JSR FPUSHO
JSR SFACTO
LDX #FTMP2
LDY #FTMP2/256
JSR FSTOR
JSR FPOPO
JSR SPERMU
LDX #FTMP2
LDY #FTMP2/256
JSR FLD1R
JSR SFDIV
JMP SROUND

; RELOAD X!

SCOSH

JSR HYP SUB
JSR SFADD
JMP DIVTWO

; COSH(X) <- (EXPE(X)+EXPE(-X))/2
; FRO <- EXPE(X), FR1 <- EXPE(-X)
; EXPE(X) + EXPE(-X)
; DIVIDE BY 2 AND RETURN

SDAY

JSR DAYSUB
BCS SCH10
JSR CLNUM

; MMDD.YYYY -> FACTOR
; ERROR => RETURN IMMEDIATELY
; CLEAR TOKBUF
; FACTOR <- FACTOR MOD 7

LDA #7
JSR INTMOD
LDA FRO+1
ASL A
ADC FRO+1
TAX
LDY #NUMLEN-3

; 0-6

; 3*(FACTOR MOD 7)

; MOVE DAY OF WEEK CHARS TO END OF TOKBUF

SDAYLP

LDA DAYTBL,X
STA TOKBUF,Y
INX
INY
CPY #NUMLEN

BNE SDAYLP
JSR DAYDSP
JMP DAYCOM

; DISPLAY DAY OF WEEK IN NUMBER LOC
; DISPLAY "***"

SDBD

JSR DAYSUB
BCS SFDON
JSR SXCHGY
JSR DAYSUB
BCS SFDON
JSR FPOP1

; DAYS BETWEEN DATES
; Y DBD X = DAYSUB(Y) - DAYSUB(X)
; ERROR => RETURN
; DAYSUB Y
; ERROR => RETURN

```

SDIV      JMP      SFSUB
          JSR      MEMSUB      ; MEM <- MEM/X
          JSR      SFDIV
          JMP      SSUM10
SDMS      LDA      #ZDMS      ; DMS -> DECIMAL DEGREES DD.MMSSSSS -> DD.DDDD
          LDX      #100      ; NUMERATOR, MOD BASE
          LDY      #60      ; DENOMINATOR
          BNE      DEGSUB      ; JMP
SDCDEG    LDA      #ZDCDEG      ; DECIMAL DEG -> DMS
          LDX      #60
          LDY      #100
DEGSUB    STX      XSAVE2
          STY      YSAVE2
          JSR      PUTMSG      ; DISPLAY MESSAGE -> DMS OR -> DECIMAL DEGREES
          INT(X) + (FRACT(X) MODFAC (1/XSAVE2))/YSAVE2 + (FRACT(X) MOD (1/XSAVE2))*XSAVE2^2/YSAVE2^2
          JSR      FMOVE
          JSR      SINTEG
          JSR      FPUSH0      ; SAVE INT(X)
          JSR      FMOVE2
          JSR      SFRACT
          JSR      FPUSH0      ; SAVE FRACT(X)
          LDA      XSAVE2
          JSR      PSET0      ; INT -> FP
          JSR      SRECIP
          JSR      SMOD      ; FRACT(X) MOD (1/XSAVE2) ALSO SETS UP MODFAC
          JSR      FPUSH0      ; SAVE MOD
          LDA      XSAVE2
          JSR      PSET0
          JSR      SSQUAR
          JSR      FPUSH0
          LDA      YSAVE2
          JSR      PSET0
          JSR      SSQUAR
          JSR      SPDIV      ; XSAVE2^2/YSAVE2^2
          JSR      SPMUL      ; MULTIPLY BY MOD
          JSR      SPADD      ; ADD INT(X)
          JSR      FPUSH0      ; SAVE RESULT ON STACK
          LDA      YSAVE2
          JSR      PSET0
          JSR      FMOVE
          JSR      FLDOM      ; LOAD MODFAC
          JSR      SFDIV      ; MODFAC/YSAVE2
          JMP      SPADD      ; ADD TO PREVIOUS RESULT & RETURN

SASINH    LDA      FRO      ; ARCSINH(X) = SIGN(X) * LN(ABSVAL(X)+SQRT(SQUARE(X)+1))
          PHA
          JSR      SABSVa      ; SAVE SIGN
          JSR      FPUSH0      ; ABSVAL(X)
          JSR      SSQUAR      ; X*X
          LDA      #1
          JSR      INTADD      ; X*X+1
          JSR      AHYPSB      ; LN(X+SQRT(X*X+1))
          PLA
          BPL      BIN100      ; RETURN
          JMP      SCHGSG      ; IF SIGN IS NEGATIVE THEN ARCSINH(X) < 0

```



```

BPL      BIN100      ;RETURN
JMP      SCHGSG      ;IF SIGN IS NEGATIVE THEN ARCSINH(X) < 0

```

COLLEEN CALCULATOR, BY C. SHAW

```

; ARCTANH(X) = (LN((1+X)/(1-X)))/2
SATANH   JSR      FPUSH0
          LDA      #1
          JSR      INTSUB      ; 1-X
          JSR      SXCHGY
          LDA      #1
          JSR      INTADD      ; 1+X
          JSR      FPOP1
          JSR      SFDIV      ; (1+X)/(1-X)
          JSR      SLN        ; LN((1+X)/(1-X))
          DIVTWO          ; MULTIPLY BY 1/2      (DIVIDE BY 2)
          LDX      #FHALF
          LDY      #FHALF/256
          JMP      LD1MUL      ; LOAD FR1 AND MULTIPLY
          ; BASE 2 OR BINARY
SBIN      LDA      #2
          JSR      SOCT10      ; CHANGE DHOFLG, STATUS MESSAGE
          LDA      BITINT
          CMP      #17
          BCC      BIN100
          LDA      #BIMSG
          JSR      ERRSUB      ; BINARY REQUIRES 16 BITS OR LESS
          LDA      #'1
          STA      TOKBUF+NUMLEN-2
          LDA      #'6
          STA      TOKBUF+NUMLEN-1
          LDA      #16
          JSR      SBITS2
BIN100    RTS
SACOSH    RTS
          ; ARCCOSH(X) = LN(X+SQRT(SQUARE(X)-1))
          JSR      FPUSH0
          JSR      SSQUAR      ; X*X
          LDA      #1
          JSR      INTSUB      ; 1-X*X
          JSR      SCHGSG      ; X*X-1
          ; FRO <- LN(TOS + SQRT(FRO))
          JSR      SSQRT      ; SQRT(X*X-1)
          JSR      FPOP1
          JSR      SFADD      ; X+SQRT(X*X-1)
          JSR      SLN        ; LN(X+SQRT(X*X-1))
          ; SET GRAD MODE
          LDA      #GRADDN
          JMP      SRADIO
          ; Y%X = (X*Y)/100      (ORDERING IS UNIMPORTANT)
          JSR      FMVPOP
          JSR      SFMUL
          LDA      FRO
          BEQ      SPE10
          DEC      FRO
          ; Y PERMU X = P(N,R) = N!/(N-R)! = Y!/((Y-X)!)
          RTS
          JSR      FMOVE
          JSR      FLD05
          JSR      SFSUB      ; Y-X
          JSR      SFACTD      ; (Y-X)!
          JSR      SXCHGY      ; LOAD Y
          JSR      SFACTD      ; Y!
          JSR      FPOPL      ; LOAD (Y-X)!

```

```

        JSR      SFDIV
        JMP      SROUND
SPRD
        JSR      MEMSUB      ; MEM ← MEM*X
        JSR      SFMUL
        JMP      SSUM10
SRAND0
        LDA      RPNALG      ; X ← RANDOM NUMBER FROM 0 TO 65535
        BNE      SRAN10
        JSR      FPUSHO      ; IF RPN PUSH PREVIOUS # AS IN SPI
SRAN10
        LDA      RANDOM
        STA      FRO
        LDA      #0
        STA      FRO+1
        JMP      IFP

SSINH
        JSR      HYP SUB      ; SINH(X) ← (EXPE(X) - EXPE(-X)) / 2
        JSR      SF SUB      ; FRO ← EXPE(X), FR1 ← EXPE(-X)
        JMP      DIV TWO      ; EXPE(X) - EXPE(-X)
                                ; DIVIDE BY 2 AND RETURN
SSUB
        JSR      MEMSUB      ; MEM ← MEM-X
        JSR      SF SUB
        JMP      SSUM10

STANH
        JSR      FPUSHO      ; TANH(X) ← SINH(X)/COSH(X)
        JSR      SSINH
        JSR      SXCHGY
        JSR      SCOSH
        JMP      SPDIV
ENDIF

```


COLLEEN CALCULATOR, BY C SHAW

```

B250          SFV          LDA      ENTFLG      ; FV = FUTURE VALUE
B250 A5 C6    BEQ          SFV05
B252 F0 1D    LDA          DUEFLG
B254 A5 C5    BPL          SFV20                ; ANNUITY => SKIP
B256 10 0B    ;                                COMPOUND INTEREST FV=PV*(1+I)^N
B258 20 DB AD    JSR        Z1IN                ; (1+I)^N
B25B A9 09    LDA          #9                  ; PV
B25D 20 E6 A3    JSR        MEMMUL              ; FRO <- FRO * MEM(A)
B260 4C 71 B2    JMP        SFV05              ; STORE NEW FV
B263          SFV20          ; ORDINARY ANNUITY FV=PMT*((1+I)^N-1)/I
;                                ; ANNUITY DUE      FV=ABOVE * (1+I)
B263 20 E1 AD    JSR        Z1INM1              ; ((1+I)^N-1)/I
B266 20 EA AD    JSR        DIVI
B269 A9 08    LDA          #8                  ; PMT
B26B 20 E6 A3    JSR        MEMMUL
B26E 20 89 A8    JSR        ZMUL1I              ; IF ANNUITY DUE THEN FRO <- FRO * (1+I)
B271 A9 05    SFV05    LDA          #5
B273 4C 4A AE    JMP        MEMSTO              ; ENTER

; COSINE ROUTINE -- ADD 90 OR PI/2 TO FRO TO DO SIN
SCOS
B276          JSR        SINMOD                ; TAKE ANGLE MOD 2*PI, 360 OR 400
B279 20 F1 A3    JSR        PIOVL              ; SET UP X & Y REGS TO LOAD PI/2 90 OR 100
B27C 20 98 DD    JSR        FLD1R              PUT PI/2 OR 90 INTO FR1
B27F 20 6A A9    JSR        SFADD              FRO=FRO + PI/2 (OR 90)
B282 4C 6D B0    JMP        SSIN2

```

DATA

THE FOLLOWING TABLES MUST NOT CROSS PAGE BOUNDARIES

**\$BA00

; DATA MUST BE AT END OF MEM

BA00 40 03 14	PICONST	BYTE	\$40, \$03, \$14, \$15, \$92, \$65 ; PI = 3.14159265
BA03 15 92 65			
BA06	SCOE		
BA06 0D 03 54		BYTE	\$BD, \$03, \$54, \$14, \$99, \$39 ; -. 00000354149939
BA09 14 99 39			
BA0C 3E 01 60		BYTE	\$3E, \$01, \$60, \$44, \$27, \$52 ; 0.000160442752
BA0F 44 27 52			
BA12 8E 46 81		BYTE	\$BE, \$46, \$81, \$75, \$43, \$55 ; -. 004681754355
BA15 75 43 55			
BA18 3F 07 96		BYTE	\$3F, \$07, \$96, \$92, \$62, \$39 ; 0.0796926239
BA1B 92 62 39			
BA1E BF 64 59		BYTE	\$BF, \$64, \$59, \$64, \$08, \$67 ; -. 6459640867
BA21 64 08 67			
BA24 40 01 57	RADPI2	BYTE	\$40, \$01, \$57, \$07, \$96, \$32 ; PI/2 = 1.570796327
BA27 07 96 32			
BA2A 40 90 00		BYTE	\$40, \$90, 0, 0, 0, 0 ; 90 (DEGREES)
BA2D 00 00 00			
		BYTE	\$41, \$01, 0, 0, 0, 0 ; 100 (GRADS)
BA30 42 06 55	C65536	BYTE	\$42, \$06, \$55, \$36, 0, 0 ; 65536 IN FP (USED IN BINFP)
BA33 36 00 00			
BA36 3F 01 74	PI0V18	BYTE	\$3F, \$01, \$74, \$53, \$29, \$25 ; PI/180 = .0174532925 DEG->RAD
BA39 53 29 25			
		BYTE	\$3F, \$01, \$57, \$07, \$96, \$33 ; PI/200 = .0157079633
BA3C 40 01 80	C1PT8	BYTE	\$40, \$01, \$80, 0, 0, 0 ; 1.8 (USED IN SCELSI)
BA3F 00 00 00			
BA42	LENGTH		
BA42 40 01 00	ONE	BYTE	\$40, \$01, 0, 0, 0, 0 ; M -> M = 1 EXACTLY
BA45 00 00 00			
BA48 3F 02 54		BYTE	\$3F, \$02, \$54, 0, 0, 0 ; INCHES->M = .0254 EXACTLY
BA4B 00 00 00			
BA4E 3F 30 48		BYTE	\$3F, \$30, \$48, 0, 0, 0 ; FEET .3048 "
BA51 00 00 00			
BA54 3F 91 44		BYTE	\$3F, \$91, \$44, 0, 0, 0 ; YARDS .9144 "
BA57 00 00 00			
BA5A 41 16 09		BYTE	\$41, \$16, \$09, \$34, \$40, 0 ; MILES 1609.344 "
BA5D 34 40 00			
BA60 3F 01 00		BYTE	\$3F, \$01, 0, 0, 0, 0 ; CM .01 "
BA63 00 00 00			
BA66 41 10 00		BYTE	\$41, \$10, 0, 0, 0, 0 ; KM 1000 "
BA69 00 00 00			
		BYTE	\$41, \$18, \$52, 0, 0, \$04 ; NAUTMI 1852.000004 ????????
BA6C	MASS		
BA6C 3F 02 83		BYTE	\$3F, \$02, \$83, \$49, \$52, \$31 ; OZ->KG = .02834952313 (NOT EXACT)
BA6F 49 52 31			
BA72 3F 45 35		BYTE	\$3F, \$45, \$35, \$92, \$37, 0 ; LB .45359237 ??
BA75 92 37 00			
BA78 3E 10 00		BYTE	\$3E, \$10, 0, 0, 0, 0 ; GM .001 EXACTLY
BA7B 00 00 00			
BA7E	VOLUME		
BA7E 3F 16 66		BYTE	\$3F, \$16, \$66, \$66, \$66, \$67 ; TSP->FLOZ = .1666666667
BA81 66 66 67			
BA84 3F 50 00		BYTE	\$3F, \$50, 0, 0, 0, 0 ; TBSP .5 EXACTLY

BA7E VOLUME
 BA7E 3F 16 66 . BYTE \$3F,\$16,\$66,\$66,\$66,\$67 ; TSP->FLOZ = .1666666667
 BA81 66 66 67
 BA84 3F 50 00 . BYTE \$3F,\$50,0,0,0,0 ; TBSP . 5 EXACTLY

COLLEEN CALCULATOR, BY C SHAW

BA87 00 00 00
 BA8A 40 08 00 . BYTE \$40,8,0,0,0,0 ; CUPS 8 "
 BA8D 00 00 00
 BA90 40 32 00 . BYTE \$40,\$32,0,0,0,0 ; QUARTS 32 "
 BA93 00 00 00
 BA96 41 01 28 . BYTE \$41,\$01,\$28,0,0,0 ; GAL 128 "
 BA99 00 00 00
 BA9C 40 33 81 . BYTE \$40,\$33,\$81,\$40,\$22,\$66 ; LITERS 33.81402266 NOT EXACT?
 BA9F 40 22 66

BAA2 20 20 20 INTCHR . BYTE " BALFV ", 'I+32, " N " ; FIRST PART OF INTEREST DISPLAY
 BAA5 42 41 4C
 BAAB 46 56 20
 BAAB 69 20 20
 BAAE 4E 20 20

BAB1 50 4D 54 PBUFF . BYTE "PMTVP " ; INTEREST DISPLAY (END OF INTCHR) AND "P" FOR PRINTER OPEN
 BAB4 50 56 20
 BAB7 4E 57 54 STACHR . BYTE "NWTN X X^2Y Y^2X*Y"
 BABA 4E 20 20
 BABD 58 20 20
 BAC0 58 5E 32
 BAC3 59 20 20
 BAC6 59 5E 32
 BAC9 58 2A 59

; SPECIAL SINGLE CHAR COMMANDS
 BACC 2A 2F 2B TOKCHR . BYTE "*/+-()=^!% ", UPAROW, DNAROW, LFAROW, RTAROW
 BACF 2D 28 29
 BAD2 3D 5E 21
 BAD5 25 1C 1D
 BAD8 1E 1F
 BADA

TOKEND
 ; TOKEN NUMBERS FOR TOKCHR COMMANDS
 BADA 86 87 88 TOKTBL . BYTE STAR, SLASH, PLUS, MINUS, LPAR, RPAR, EQUAL, POWER, FACTOR, MOD
 BADD 89 8A 8B
 BAE0 8C 51 25
 BAE3 41

; BSTEP, SSTEP, DELETE, INSERT ARE PART OF BOTH TOKTBL & SPCTBL
 ; SPECIAL COMMANDS IN STORE PROGRAM MODE (EXECUTED IMMEDIATELY, NOT STORED)
 BAE4 0A 6C 1F SPCTBL . BYTE BSTEP, SSTEP, DELETE, INSERT, CLPROG, ZEND, PROGRAM, LIST, SAVE, LOAD, RST
 BAE7 34 12 20
 BAEA 53 3A 65
 BAED 3C 63
 BAEF

SPCEND
 BAEF 4B KBUFF . BYTE "K" ; K FOR KEYBOARD OPEN
 BAF0 20 2A 2A STARMS . BYTE " ***"
 BAF3 2A

; GRAPHICS CHARS FOR SCREEN DISPLAY -64 => CONTROL KEY HIT (USED IN PTLIN1)
 BAF4 11 17 05 CHRTAB . BYTE 'Q-64, 'W-64, 'E-64, 'A-64, 'S-64, 'D-64, 'Z-64, 'X-64, 'C-64
 BAF7 01 13 04
 BAFB 1A 18 03

BAFD 58 59 32 CHTAB2 . BYTE "XY23456789" STACK LABELS
 BB00 33 34 35
 BB03 36 37 38
 BB06 39

; COM, BAL, BAH, AX1 FOR IOCB
 V BB07 03 EF BA CIOTAB . BYTE OPEN, KBUFF, KBUFF/256, INPUT ; OPEN K: FOR INPUT
 BB0A 04
 V BB0B 03 B1 BA . BYTE OPEN, PBUFF, PBUFF/256, OUTPUT ; OPEN P: FOR OUTPUT
 BB0E 08
 V BB0F 03 00 05 . BYTE OPEN, TOKBUF, TOKBUF/256 ; OPEN TIOCB

8812' 00

BYTE CLOSE

; CLOSE (OTHER PARAMS DON'T MATTER)

COLLEEN CALCULATOR, BY C SHAW

```

      IF      ASMBL      , DON'T ASSEMBLE
      MY EXPERIMENTAL SCOE FOR SIN,COS      (9 TERMS INSTEAD OF 6)

; EXPSC
      BYTE    $3A,$06,$06,$69,$35,$73 ; 6.066935731E-12
      BYTE    $B8,$06,$68,$80,$35,$12 ; -6.688035123E-10 = -(PI/2)^15/15!
      BYTE    $3C,$05,$69,$21,$72,$92 ; 5.692172922E-08 = (PI/2)^13/13!
      BYTE    $B8,$03,$59,$88,$43,$24 ; -.00000359884324 - (PI/2)^11/11!
      BYTE    $3E,$01,$60,$44,$11,$85 ; 0.000160441185
      BYTE    $DE,$46,$81,$75,$41,$35 ; -.004681754155
      BYTE    $3F,$07,$96,$92,$62,$62 ; 0.0796926262
      BYTE    $BF,$64,$59,$64,$09,$75 ; -.6459640975
      BYTE    $40,$01,$57,$07,$96,$32 ; PI/2 = 1.570796327

; C10000
      BYTE    $42,$01,0,0,0,0 ; 10000 (USED IN DAY CALCULATIONS)
; C365
      BYTE    $41,$03,$65,0,0,0 ; 365
; CPT75
      BYTE    $3F,$75,0,0,0,0 ; .75 = 3/4
DEGREE
      BYTE    $3F,$90,0,0,0,0 ; 180/200 = .9 GRAD -> DEG
      BYTE    $40,$57,$29,$57,$79,$51 ; 180/PI = 57.29577951 RAD -> DEG
; CFT
      BYTE    $3F,$30,$48,0,0,0 ; FT->M .3048 EXACTLY
; CMI
      BYTE    $40,$01,$60,$93,$44,0 ; MI->KM 1.609344 EXACTLY
; CLB
      BYTE    $3F,$45,$35,$92,$37,0 ; LB->KG .45359237 ??
; CL
      BYTE    $3F,$26,$41,$72,$05,$24 ; L->GAL .2641720524
; ONE
      BYTE    $40,$01,0,0,0,0 ; ONE

      MY EXPERIMENTAL P10COF FOR EXP FUNCTION

; P10COF
      BYTE    $3D,$09,$79,$28,$29,$75 ; .000009792829753 = (LN(10)/2)^9/9!
      BYTE    $3D,$76,$55,$34,$94,$63 ; .00007655349463 = (LN(10)/2)^8/8!
      BYTE    $3E,$05,$31,$94,$81,$65 ; .000531948165
      BYTE    $3E,$32,$34,$31,$01,$36 ; .003234310136
      BYTE    $3F,$01,$68,$55,$71,$65 ; .0168557165
      BYTE    $3F,$07,$32,$03,$44,$68 ; .0732034468 = (LN(10)/2)^4/4!
      BYTE    $3F,$25,$43,$34,$82,$44 ; .2543348244
      BYTE    $3F,$66,$27,$37,$26,$38 ; .6627372638 = (LN(10)/2)^2/2!
      BYTE    $40,$01,$15,$12,$92,$55 ; 1.15129255 = LN(10)/2
      BYTE    $3F,$99,$99,$99,$99,$99 ; .9999999999 APPROX. 1

      LENGTH OF EACH MONTH + 1 IN BCD
; MAXDAY
      BYTE    1+$31,$30,1+$31,1+$30,1+$31,1+$30,1+$31,1+$30,1+$31,1+$30,1+$31
; DAYIRM
      BYTE    0,0,3,3,4,4,5,5,6,6,7 ; # OF DAYS LESS THAN 31/MONTH FOR EACH MONTH
; DAYTBL
      BYTE    "SATSUMONTUEWEDTHUFRI"
      ENDIF

```

BB13	49 B2 9C	JMPTBL	
BB16	A4 18 A7	. WORD	SABSV, SACOS, SADV, SALG, SALGN, SAND, SASIN, SATAN, SBAL, SBITS
BB19	79 A7 7D		
BB1C	A7 CB AB		
BB1F	05 B1 1B		
BB22	B1 18 AE		
BB25	E0 A4		
BB27	90 A9 40	. WORD	SBSTEP, SC, SCALL, SCDEG, SCHGSG, SCLCAL, SCLINI, SCLMEM, SCLPRD, SCLR, SCLSTAT, SCLX
BB2A	A9 9F AB		
BB2D	BC AD 82		
BB30	A5 84 AB		
BB33	8B A5 C9		
BB36	A5 F5 A9		
BB39	74 A7 9B		
BB3C	A5 B0 A1		
BB3F	41 AD E3	. WORD	SCM, SCMPND, SCOMPL, SCONTI, SCOS, SCRAD, SCUP, SDEC, SDEG, SDELET
BB42	AD 7F A5		
BB45	F0 BC 76		
BB48	B2 AD AD		
BB4B	69 AD 04		
BB4E	A7 53 A7		
BB51	FE AB		
BB53	93 AA 0B	. WORD	SEND, SENTER, SEXPE, SEXPT, SF, SFACTD, SFIND, SFIX, SFLOZ, SFRACT, SFT
BB56	AE 00 9B		
BB59	07 9B CB		
BB5C	AD E9 A5		
BB5F	0F AE 5D		
BB62	A5 5D AD		
BB65	7A A9 35		
BB68	AD		
BB69	50 B2 F7	. WORD	SEV, SEVDUE, SEVDOR, SGAL, SGM, SGOTO, SHEX, SI, SIN, SINER, SINTEG, SKG
BB6C	AD FB AD		
BB6F	71 AD 57		
BB72	AD 40 AB		
BB75	0B A7 54		
BB78	AE 31 AD		
BB7B	34 AC B3		
BB7E	A6 4B AD		
BB81	45 AD 75	. WORD	SKM, SL, SLB, SLIST, SLN, SLOAD, SLOGTE, SLSHF, SM, SMI
BB84	AD 53 AD		
BB87	22 AA B3		
BB8A	A6 C7 AC		
BB8D	BE A6 DA		
BB90	A7 2D AD		
BB93	3D AD		
BB95	E8 A6 9B	. WORD	SMOD, SN, SNOP, SNOTRA, SNWEIG, SOCT, SOFF, SON, SOR, SOZ, SPAUSE
BB98	AE 92 AA		
BB9B	DC BC 63		
BB9E	B0 0C A7		
BBA1	2E A7 37		
BBA4	A7 CC AB		
BBA7	4F AD 4B		
BBAA	AA		
BBAB	BD A4 1E	. WORD	SPI, SPMT, SPOLAR, SPOP, SPOPC, SPOWER, SPRINT, SPROGR, SPUSH, SPV
BBAE	AF 7E B1		
BBB1	9D 9F 8B		
BBB4	AB 19 A6		
BBB7	0B A9 66		
BBBA	AA BB 9F		

COLLEEN CALCULATOR, BY C SHAW

BBBD 66 AF	
BBBF 03 AE 07	. WORD
BBC2 AE 6D AD	SPVDUE, SPVORD, SGT, SCORRE, SRAD, SRCL, SRECIP, SRECTA, SRETUR, SROOT
BBC5 F5 AF 57	
BBC8 A7 7F 9F	
BBCB 24 A9 1D	
BBCE 9F E6 AB	
BBD1 16 A6	
BBD3 60 A6 B1	. WORD
BBD6 A7 DE A7	SROUND, SRPN, SRSHF, SRESET, SRUN, SSAVE, SSIN, SSLOPE, SSMINU, SSPLUS, SSGRT
BBD9 8A AA ED	
BBDC BC DF AC	
BBDF 6A B0 3A	
BBE2 B0 AB BF	
BBE5 AF BF A7	
BBE8 B1	
BBE9 5A AB CC	. WORD
BBEC A9 60 AB	SSQUAR, SSSTEP, SSTO, SSTP, SSUM, STAN, STBSP, STRACE, STRUNC, STSP, SX
BBEF 93 AA A7	
BBF2 AB 2B A9	
BBF5 65 AD E0	
BBF8 BC 7D A6	
BBFB 61 AD 0A	
BBFE B0	
BBFF E8 9F B3	. WORD
BC02 AB 15 AB	SXCHGY, SXCHM, SXEQ, SXGE, SXLT, SXMEAN, SXNE, SXOR, SXSTDD, SXVARI
BC05 23 AB 2E	
BC08 AB D9 AF	
BC0B 39 AB D0	
BC0E AB EC BF	
BC11 E1 AF	
BC13 5B A9 39	. WORD
BC16 AD 1C B0	SY, SYD, SYINTE, SYMEAN, SYSTDD, SYVARI
BC19 DD AF F2	
BC1C BF E5 AF	
BC1F 94 AB 37	. WORD
BC22 A9 67 A9	SPMUL, SPDIV, SPADD, SPSUB
BC25 B0 A9	

OUTPUT FROM BASIC PROGRAM DK1: WORDSG.BAS
 TABLE BYTE "TEROANCSLPDIMUFGXYVHBKWQZJ"

BC27 20 54 45
 BC2A 52 4F 41
 BC2D 4E 43 53
 BC30 4C 50 44
 BC33 49 4D 55
 BC36 46 47 58
 BC39 59 56 48
 BC3C 42 4B 57
 BC3F 51 5A 4A
 BC42

ERRTBL

TWO OPS IN A ROW

BC42 12 20 85
 BC45 15 B9 1D
 BC48 71 61 45
 BC4B 08

TOPMSG . BYTE

18, 32, 133, 21, 185, 29, 113, 97, 69, 8

10/16

BC4C 1D 75 21
 BC4F 04 6A DC
 BC52 18 5E E6
 BC55 7C 15 41
 BC58 7F E0 63
 BC5B 40

KEYMSG . BYTE

NOT VALID COMMAND OR NUMBER

29, 117, 33, 4, 106, 220, 24, 94, 230, 124, 21, 65, 127, 224, 99, 64

16

27

BC5C 16 05 30
 BC5F 20 F2 F5
 BC62 82 15 04
 BC65 40 0A 08

BOMSG . BYTE

HEX/OCT OVRFLW

22, 5, 48, 32, 242, 245, 130, 21, 4, 64, 10, 8

18/14

BC68 15 7F E0
 BC6B 63 41 92
 BC6E 68 07 13
 BC71 EB 20 30

NSEMSG . BYTE

NUMBER STACK EMPTY

21, 127, 224, 99, 65, 146, 104, 7, 19, 235, 32, 48

BC74 14 7F E0
 BC77 63 41 92
 BC7A 68 07 10
 BC7D 0F AA

NSFMSG . BYTE

NUMBER STACK FULL

20, 127, 224, 99, 65, 146, 104, 7, 16, 15, 170

BC7F 10 5B 19
 BC82 26 80 71
 BC85 3E B2 03

OSEMSG . BYTE

OP STACK EMPTY

16, 91, 25, 38, 128, 113, 62, 178, 3

BC88 0F 5B 19
 BC8B 26 80 71
 BC8E 00 FA A0

OSFMSG . BYTE

OP STACK FULL

15, 91, 25, 38, 128, 113, 0, 250, 160

BC91 16 7F E0
 BC94 63 41 5F
 BC97 21 50 01
 BC9A 46 70 13

BITMSG . BYTE

NUMBER OUT OF RANGE

22, 127, 224, 99, 65, 95, 33, 80, 1, 70, 112, 19

BC9D 15 25 51
 BCA0 E6 70 31
 BCA3 80 56 46
 BCA6 82 34 90

DIGMSG . BYTE

TOO MANY CHARACTERS

21, 37, 81, 230, 112, 49, 128, 86, 70, 130, 52, 144

BCA9 17 64 D2
 BCAC 05 E3 2D

CRYMSG . BYTE

ARITHMETIC OVERFLOW

23, 100, 210, 5, 227, 45, 129, 80, 67, 64, 10, 80, 128

13/19

BCA6 82 34 0
BCA9 17 64 D2
BCAC 05 E3 2D

CRYMSG . BYTE

ARITHMETIC OVERFLOW
23, 100, 210, 5, 227, 45, 129, 80, 67, 64, 10, 80, 128

COLLEEN CALCULATOR, BY C SHAW

BCAF 81 50 43
BCB2 40 0A 50
BCB5 80

; END OF PROG MEM
EPMSG . BYTE 17, 55, 193, 80, 1, 180, 80, 17, 227, 224

BCB6 11 37 C1
BCB9 50 01 B4
BCBC 50 11 E3
BCBF E0

; CALL STACK EMPTY
CLEMSG . BYTE 18, 134, 170, 25, 38, 128, 113, 62, 178, 3

BCC0 12 86 AA
BCC3 19 26 80
BCC6 71 3E B2
BCC9 03

; CALL STACK FULL
CLFMSG . BYTE 17, 134, 170, 25, 38, 128, 113, 0, 250, 160

BCCA 11 86 AA
BCCD 19 26 80
BCD0 71 00 FA
BCD3 A0

; UNIT MISMATCH
UNIMSG . BYTE 14, 247, 210, 30, 217, 230, 40, 5

BCD4 0E F7 D2
BCD7 1E D9 E6
BCDA 28 05
BCDC

SNOTRACE
LDA #0 ; TRACE OFF 20 LINES
BEQ STR10

BCDC A9 00
BCDE F0 02
BCE0

STRACE
LDA #1 ; TRACE ON
STR10

BCE0 A9 01
BCE2

STA TRACE
LDX PRDG ; PROGRAM IN EXECUTION?
BEQ STR20 ; NO.

BCE2 B5 BC
BCE4 A6 BB
BCE6 F0 04
BCE8

STR15
EOR #\$01 ; YES. TRACE DETERMINES DSPFLG
STA DSPFLG

BCE8 49 01
BCEA B5 BD
BCEC

STR20
RTS
JSR SRESET ; GOTO 0 AND RUN
SCONTI ; CONTINUE=> RUN STARTING AT CURRENT PC

BCEC 60
BCED 20 8A AA
BCF0

LDX #EXEC
STX PRDG
LDA TRACE
BPL STR15 ; JMP IF NOTRACE THEN DSPFLG<-1

BCF0 A2 02
BCF2 86 BB
BCF4 A5 BC
BCF6 10 F0

```

***-1/256+1*256          ; GOTO NEXT PAGE BOUNDARY
;
BD00 29 37 23      ;
BD03 41 84 50      ; PROMSG . BYTE      ENTER PRG ADDR 0-1023
BD06 11 6C C4      ;          41, 55, 35, 65, 180, 80, 17, 108, 196, 16, 243, 0, 242, 208, 243, 16, 243, 0, 243, 32, 243, 48
BD09 10 F3 00
BD0C F2 D0 F3
BD0F 10 F3 00
BD12 F3 20 F3
BD15 30

BD16 12 37 23      ;
BD19 41 0E 30      ; FIXMSG . BYTE      ENTER 0-8
BD1C 0F 2D 0F      ;          18, 55, 35, 65, 15, 48, 15, 45, 15, 56
BD1F 38

BD20 16 37 23      ;
BD23 41 0F 31      ; BTMSG . BYTE      ENTER 1-32
BD26 0F 2D 0F      ;          22, 55, 35, 65, 15, 49, 15, 45, 15, 51, 15, 50
BD29 33 0F 32

BD2C 1B 37 23      ;
BD2F 41 43 01      ; MEMMSG . BYTE      ENTER REG 0-99
BD32 10 F3 00      ;          27, 55, 35, 65, 67, 1, 16, 243, 0, 242, 208, 243, 144, 243, 144
BD35 F2 D0 F3
BD38 90 F3 90

BD3B 0F 37 23      ;
BD3E 41 00 DA      ; FSPMSG . BYTE      ENTER FILESPEC
BD41 39 B3 80      ;          15, 55, 35, 65, 0, 218, 57, 179, 128

BD44 13 37 23      ;
BD47 41 C3 9D      ; CN2MSG . BYTE      ENTER DESIRED UNITS
BD4A 43 C1 F7      ;          19, 55, 35, 65, 195, 157, 67, 193, 247, 210, 144
BD4D D2 90

BD4F 14 85 70      ;
BD52 43 49 D5      ; CN3MSG . BYTE      CONVERSION COMPLETE
BD55 71 85 E8      ;          20, 133, 112, 67, 73, 213, 113, 133, 235, 163, 35
BD58 A3 23

BD5A 05 25 10      ;
BD5D 00      ; CELMSG . BYTE      TO F
;          5, 37, 16, 0

BD5E 04 25 18      ;
;          TO C
;          4, 37, 24
;          TO POLAR Y, X->Y=ANGLE, X=RADIUS
BD61 36 25 1B      ; ZRECT . BYTE      54, 37, 27, 90, 100, 17, 3, 15, 44, 2, 15, 45, 15, 62, 3, 15, 61, 103, 1, 163, 15, 44, 2, 15, 61, 70, 205, 249
BD64 5A 64 11
BD67 03 0F 2C
BD6A 02 0F 2D
BD6D 0F 3E 03
BD70 0F 3D 67
BD73 01 A3 0F
BD76 2C 02 0F
BD79 3D 46 CD
BD7C F9

BD7D 35 25 14      ;
BD80 38 21 10      ; ZPOLAR . BYTE      TO RECT Y=ANGLE, X=RADIUS->Y, X
;          53, 37, 20, 56, 33, 16, 48, 243, 214, 112, 26, 48, 242, 192, 32, 243, 212, 108, 223, 144, 242, 208, 243, 224, 48, 24

```


BD7D 35 25 14
BD80 38 21 10

ZPOLAR . BYTE

53, 37, 20, 56, 33, 16, 48, 243, 214, 112, 26, 48, 242, 192, 32, 243, 212, 108, 223, 144, 242, 208, 243, 224, 48, 24

COLLEEN CALCULATOR, BY C SHAW

BD83 30 F3 D6
BD86 70 1A 30
BD89 F2 C0 20
BD8C F3 D4 6C
BD8F DF 90 F2
BD92 D0 F3 E0
BD95 30 F2 C0
BD98 20

BD99 06 25 14
BD9C 6C

. ZDEG . BYTE

TO RAD
6, 37, 20, 108

BD9D 07 25 1C
BDA0 30 10

. ZRAD . BYTE

TO DEG
7, 37, 28, 48, 16

BDA2 04 25 1E

. ZM . BYTE

TO M
4, 37, 30

BDA5 07 25 10
BDAB 70 10

. ZKG . BYTE

TO KG
7, 37, 16, 112, 16

BDAA 0A 25 10
BDAD 0A 15 0A

. ZFL . BYTE

TO FL_OZ
10, 37, 16, 10, 21, 10

BDB0 0A 34 45
BDB3 41 0F 2D

. ERRMSG . BYTE

ERROR -
10, 52, 69, 65, 15, 45

BDB6 39 0F 7D
BDB9 11 11 62
BDBC 64 D1 86
BDBF AB FA 62
BDC2 54 18 5B
BDC5 03 4D 01
BDC8 05 21 0F
BDCB 31 0F 39
BDCE 0F 37 0F
BDD1 39 11 10

. STATLN . BYTE

[ATARI CALCULATOR COPYRIGHT 1979
57, 15, 125, 17, 17, 98, 100, 209, 134, 168, 250, 98, 84, 24, 91, 3, 77, 1, 5, 33, 15, 49, 15, 57, 15, 55, 15, 57, 17, 1

BDD4 38 16 A0
BDD7 11 14 6C
BDDA 1C 38 10
BDDD 6D 29 0F
BDE0 31 0F 36
BDE3 10 0D 02
BDE6 0F 38 10
BDE9 00 4C F3
BDEC 13 72 34
BDEF 0F 9B

. STLN2 . BYTE

ALG RAD DEC BITS16 FIX8 FVDUE ENTER1
56, 22, 160, 17, 20, 108, 28, 56, 16, 109, 41, 15, 49, 15, 54, 16, 13, 2, 15, 56, 16, 0, 76, 243, 19, 114, 52, 15, 155

BDF1 31 0F 7C
BDF4 11 11 11
BDF7 92 68 07
BDFA 11 11 10
BDFD F7 C1 11
BE00 11 43 01
BE03 D9 23 49
BE06 11 11 10
BE09 F7 C0
BE0B

. STKLIN . BYTE

STACK
REGISTERS
49, 15, 124, 17, 17, 17, 146, 104, 7, 17, 17, 16, 247, 193, 17, 17, 67, 1, 217, 35, 73, 17, 17, 16, 247, 192

KEYWRD

COLLEEN CALCULATOR, BY C SHAW

BE0B 46 06	. BYTE	70, 6	ACOS	1
BE0D 94 6B 59	. BYTE	148, 104, 89	ADV	2
BE10 46 C0	. BYTE	70, 192	ALG	3
BE12 44 6A 01	. BYTE	68, 106, 1	ALGN	4
BE15 56 A0 17	. BYTE	86, 160, 23	AND	5
BE18 36 7C	. BYTE	54, 124	ASIN	6
BE1A 46 9D	. BYTE	70, 157	ATAN	7
BE1C 74 62 67	. BYTE	116, 98, 103	BAL	8
BE1F 40 66	. BYTE	64, 102	BITS	9
BE21 A5 06 D2	. BYTE	165, 6, 210	BST	10
BE24 94 06 92	. BYTE	148, 6, 146	C	11
BE27 18	. BYTE	24	CALL	12
BE28 48 6A	. BYTE	72, 106	CDEQ	13
BE2A A5 8C 30	. BYTE	165, 140, 48	CHGSGN	14
BE2D 19 80 50	. BYTE	25, 128, 80, 25, 1	CLCALL	15
BE30 19 01	. BYTE		CLINT	16
BE32 76 BA 86	. BYTE	118, 138, 134, 170	CLMEM	17
BE35 AA	. BYTE		CLPROG	18
BE36 58 AD 72	. BYTE	88, 173, 114	CLR	19
BE39 58 AE 3E	. BYTE	88, 174, 62	CLSTAT	20
BE3C 78 AB 45	. BYTE	120, 171, 69, 1	CLX	21
BE3F 01	. BYTE		CM	22
BE40 38 A4	. BYTE	56, 164	CMPND	23
BE42 68 A9 26	. BYTE	104, 169, 38	COMP	24
BE45 24 8A 02	. BYTE	36, 138, 2	CONT	25
BE48 28	. BYTE	40	COS	26
BE49 E5 8E B7	. BYTE	229, 142, 183	CRAD	27
BE4C C4 85 EB	. BYTE	196, 133, 235	CUP	28
BE4F 48 57	. BYTE	72, 87		
BE51 23 85	. BYTE	35, 133		
BE53 94 84 6C	. BYTE	148, 132, 108		
BE56 38 FB	. BYTE	56, 251		

COLLEEN CALCULATOR, BY C SHAW

BE58 3C 38	. BYTE	60, 56	DEC	29
BE5A 4C 30	. BYTE	76, 48	DEG	30
BE5C 13 C3	. BYTE	19, 195	DEL	31
BE5E A3 37	. BYTE	163, 55	END	32
BE60 C5 37 23	. BYTE	197, 55, 35	ENTER	33
BE63 45 30 2B	. BYTE	69, 48, 43	EXPE	34
BE66 37 30 2B BE69 23	. BYTE	55, 48, 43, 35	EXPTEN	35
BE6A 72 00	. BYTE	114, 0	F	36
BE6C 50 06 82	. BYTE	80, 6, 130	FACT	37
BE6F 50 0D 7C	. BYTE	80, 13, 124	FIND	38
BE72 50 0D 02	. BYTE	80, 13, 2	FIX	39
BE75 60 0A 50	. BYTE	96, 10, 80	FLOZ	40
BE78 A5 00 46	. BYTE	165, 0, 70	FRAC	41
BE7B 83 00	. BYTE	131, 0	FT	42
BE7D 24 00 04	. BYTE	36, 0, 4	FV	43
BE80 70 00 4C BE83 F3	. BYTE	112, 0, 76, 243	FVDUE	44
BE84 70 00 45 BE87 4C	. BYTE	112, 0, 69, 76	FVORD	45
BE88 40 16	. BYTE	64, 22	GAL	46
BE8A A3 01	. BYTE	163, 1	GM	47
BE8C E5 01 52	. BYTE	229, 1, 82	GOTO	48
BE8F 55 05 30	. BYTE	85, 5, 48	HEX	49
BE92 21	. BYTE	33	I	50
BE93 D2 D7	. BYTE	210, 215	IN	51
BE95 3D 79	. BYTE	61, 121	INS	52
BE97 3D 72	. BYTE	61, 114	INT	53
BE99 40 70	. BYTE	64, 112	KG	54
BE9B 13 07	. BYTE	19, 7	KM	55
BE9D E1	. BYTE	225	L	56
			LB	57

BEVE A3 A0	BYTE	163, 160	LIST	58
BEA0 64 AD 92	BYTE	100, 173, 146	LN	59
BEA3 2A	BYTE	42	LOAD	60
BEA4 74 A5 6C	BYTE	116, 165, 108	LOGTEN	61
BEA7 7A 50 12 BEAA 37	BYTE	122, 80, 18, 55	LSHF	62
BEA8 6A 90 50	BYTE	106, 144, 80	M	63
BEAE 01	BYTE	1	MI	64
BEAF E2 ED	BYTE	226, 237	MOD	65
BEB1 3E 5C	BYTE	62, 92	N	66
BEB3 17	BYTE	23	NOP	67
BEB4 37 5B	BYTE	55, 91	NOTRC	68
BEB6 57 52 48	BYTE	87, 82, 72	NWT	69
BEB7 47 08	BYTE	71, 8	OCT	70
BEBB 23 58	BYTE	35, 88	OFF	71
BEBD 25 50 00	BYTE	37, 80, 0	ON	72
BECO 02 57	BYTE	2, 87	OR	73
BEC2 25	BYTE	37	OZ	74
BEC3 43 50	BYTE	67, 80	PAUSE	75
BEC5 A5 B6 F9	BYTE	165, 182, 249	PI	76
BEC8 32 BD	BYTE	50, 189	PMT	77
BECA 3B E2	BYTE	59, 226	POLAR	78
BECC 5B 5A 64	BYTE	91, 90, 100	POP	79
BECF 3B 5B	BYTE	59, 91	POPC	80
BED1 4B 5B	BYTE	75, 91	POWER	81
BED3 86 B5 0B BED6 34	BYTE	134, 181, 8, 52	PRINT	82
BED7 5B 4D 72	BYTE	91, 77, 114	PROG	83
BEDA 5B 45 01	BYTE	91, 69, 1	PUSH	84
BEDD 5B F9 05	BYTE	91, 249, 5	PV	85
BEE0 3B 04	BYTE	59, 4	PVDUE	86

COLLEEN CALCULATOR, BY C SHAW

BEE2 6B 04 CF	. BYTE	107, 4, 207		
BEE5 36 B0 45	. BYTE	54, 176, 69, 76	PVORD	87
BEE8 4C				
BEE9 30 92	. BYTE	48, 146	QT	88
BEEB 14	. BYTE	20	R	89
BEEC 34 6C	. BYTE	52, 108	RAD	90
BEEE 34 8A	. BYTE	52, 138	RCL	91
BEF0 54 38 DB	. BYTE	84, 56, 219	RECIP	92
BEF3 44 38	. BYTE	68, 56	RECT	93
BEF5 26 43 2F	. BYTE	38, 67, 47, 71	RETURN	94
BEF8 47				
BEF9 44 55	. BYTE	68, 85	ROOT	95
BEFB 25 45 F7	. BYTE	37, 69, 247	ROUND	96
BEFE C3 4B	. BYTE	195, 75	RPN	97
BF00 76 49 05	. BYTE	118, 73, 5, 0	RSHE	98
BF03 00				
BF04 34 92	. BYTE	52, 146	RST	99
BF06 34 F7	. BYTE	52, 247	RUN	100
BF08 59 60 43	. BYTE	89, 96, 67	SAVE	101
BF0B 39 D7	. BYTE	57, 215	SIN	102
BF0D 59 A5 B3	. BYTE	89, 165, 179	SLOPE	103
BF10 69 ED 7F	. BYTE	105, 237, 127	SMINUS	104
BF13 95 9B AF	. BYTE	149, 155, 175	SPLUS	105
BF16 95 90 94	. BYTE	149, 144, 148	SQRT	106
BF19 27 90 9F	. BYTE	39, 144, 159, 100	SQUARE	107
BF1C 64				
BF1D 33 99	. BYTE	51, 153	SST	108
BF1F 23 92	. BYTE	35, 146	STO	109
BF21 53 92	. BYTE	83, 146	STP	110
BF23 B3 9F	. BYTE	179, 159	SUM	111
BF25 E3 26	. BYTE	227, 38	TAN	112
BF27 75 20 69	. BYTE	117, 32, 105	TBSP	113
			TRACE	114

BF2A B5 24 68	. BYTE	181, 36, 104	TRUNC	115
BF2D 35 24 F7	. BYTE	53, 36, 247	TSP	116
BF30 B3 29	. BYTE	131, 41	X	117
BF32 B2 02	. BYTE	178, 2	XCHGY	118
BF34 90 28 05 BF37 01 03	. BYTE	144, 40, 5, 1, 3	XCHM	119
BF39 60 28 05	. BYTE	96, 40, 5	XEQ	120
BF3C E5 02 30	. BYTE	229, 2, 48	XGE	121
BF3F 95 02 01	. BYTE	149, 2, 1	XLT	122
BF42 34 02 A2	. BYTE	52, 2, 162	XMEAN	123
BF45 60 2E 36	. BYTE	96, 46, 54	XNE	124
BF48 74 02 73	. BYTE	116, 2, 115	XOR	125
BF4B 40 25	. BYTE	64, 37	XSD	126
BF4D 44 02 9C	. BYTE	68, 2, 156	XVAR	127
BF50 60 20 46	. BYTE	96, 32, 70	Y	128
BF53 42 03	. BYTE	66, 3	YD	129
BF55 30 3C	. BYTE	48, 60	YINT	130
BF57 50 3D 72	. BYTE	80, 61, 114	YMEAN	131
BF5A 60 3E 36	. BYTE	96, 62, 54	YSD	132
BF5D 74 03 9C	. BYTE	116, 3, 156	YVAR	133
BF60 60 30 46 BF63 40	. BYTE	96, 48, 70, 64		
0086	STAR =	134		
0051	POWER =	81		
0025	FACTOR =	37		
0041	MOD =	65		
000A	BSTEP =	10		
0012	CLPROG =	18		
0020	ZEND =	32		
0053	PROGRAM =	83		
006C	SSTEP =	108		
006E	STP =	110		
0034	INSERT =	52		
001F	DELETE =	31		
003A	LIST =	58		
0065	SAVE =	101		
003C	LOAD =	60		
0021	ENTER =	33		
0063	RST =	99		
BF64	PRIDTB			

BF64 DD EE E6	. BYTE	221, 238, 230, 221, 237, 237, 237, 222, 238, 237	ACOS	0
BF67 DD ED ED				
BF6A ED DE EE				
BF6D ED				
BF6E ED EE DE	. BYTE	237, 238, 222, 221, 237, 238, 238, 221, 221, 237	CLX	0
BF71 DD ED EE				
BF74 EE DD DD				
BF77 ED				
BF78 ED EE EE	. BYTE	237, 238, 238, 238, 237, 222, 237, 238, 238, 237	FRAC	0
BF7B EE ED DE				
BF7E ED EE EE				
BF81 ED				
BF82 ED AE EA	. BYTE	237, 174, 234, 238, 238, 222, 229, 238, 222, 221	LOGTEN	0
BF85 EE EE DE				
BF88 E5 EE DE				
BF8B DD				
BF8C E9 EE EE	. BYTE	233, 238, 238, 238, 237, 237, 221, 233, 222, 174	POWER	0
BF8F EE ED ED				
BF92 DD E9 DE				
BF95 AE				
BF96 EE DD DD	. BYTE	238, 221, 221, 221, 238, 237, 222, 237, 221	SAVE	0
BF99 DD EE ED				
BF9C DE ED ED				
BF9F DD				
BFA0 DD DD D5	. BYTE	221, 221, 213, 221, 222, 221, 221, 136, 119, 34, 16	XGE	0
BFA3 DD DE DD				
BFA6 DD 88 77				
BFA9 22 10				
BFAB	SSMINU			
BFAB A9 01	LDA	#1		; SIGMA MINUS (DELETE PREVIOUS ENTRY)
BFAD D0 02	BNE	SIGSUB		; JMP
BFAF	SSPLUS			
BFAF A9 00	LDA	#0		; SIGMA PLUS: ADD NEW X,Y PAIR
BFBI	SIGSUB			
BFBI B5 C8	STA	MEMFLG		; THIS PART IS COMMON TO BOTH SSMINU AND SSPLUS
BFBI 20 BB 9F	JSR	FPUSH0		
BFBI A9 01	LDA	#1		
BFBI 20 B9 A1	JSR	PSET0		; N <- N+1
BFBI A9 04	LDA	#4		
BFBI 20 AA AF	JSR	MEMADD		
BFC0 A9 05	LDA	#5		; SIGMA X
BFC2 20 97 AF	JSR	ZSIGMA		; COMPUTE SIGMA X, SIGMA X SQUARED
BFC5 20 86 9F	JSR	FPOP1		; LOAD X
BFC8 20 18 9F	JSR	FLDOS		; LOAD Y (& LEAVE ON STACK)
BFCB 20 97 AB	JSR	SFMUL		
BFCE A9 09	LDA	#9		; SIGMA (X*Y)
BFDO 20 AA AF	JSR	MEMADD		
BFDO A9 07	LDA	#7		; SIGMA Y
BFDO 20 97 AF	JSR	ZSIGMA		; COMPUTE SIGMA Y, SIGMA Y SQUARED
BFDB 4C 9D 9F	JMP	FPOPO		; Y -> X

BFDB		SINMOD	JSR	FPUSHO	; FIND ANGLE MOD 2*PI, 360 OR 400
BFDB 20 BB 9F			JSR	PIOVL	; DEPENDING ON CURRENT MODE. SAVE FRO DN STACK FOR MOD
BFDE 20 F1 A3			JSR	FLDOR	; LOAD PI/2, 90, OR 100
BFE1 20 89 DD			LDA	#4	
BFE4 A7 04			JSR	INTMUL	; MULTIPLY BY 4 (LOSE ACCURACY IN 10TH DIGIT OF 2*PI)
BFE6 20 84 A8			JMP	SMOD	; TAKE MOD AND RETURN
BFE7 4C E8 A6			JSR	SXVARI	; STANDARD DEVIATION (X) <- SQRT(VARIANCE(X))
BFEC 20 E1 AF	SXSTDD		JMP	SSQRT	
BFEF 4C A7 B1			JSR	SYVARI	; STDDEV(Y) <- SQRT(VAR(Y))
BFF2 20 E5 AF	SYSTDD		JMP	SSQRT	
BFF5 4C A7 B1					
		== \$A000 + \$2000 - 6			; CARTRIDGE START INFO
BFFA 4A 98		. WORD	START		; COLD/WARM START ADDRESS
BFFC 00 05		. BYTE	0, 4+1		; BOOT DISK & RUN CARTRIDGE
BFFE B4 9B		. WORD	INIT		; POWER UP START VECTOR
		. END			

COLLEEN CALCULATOR, BY C SHAW

SYMBOL TABLE							
AFP	D800	ALGNOP	0002	ALGP	0001	ARCSUB	9FBC
ASAVE	0562	ASMBL	0000	ATAN1	B13E	ATAN2	B172
ATCOEF	DFAE	ATNOUT	B17B	AUDC1	D201	AUDF1	D200
BACKSP	007E	BB10	A45A	BB30	A469	BBLP1	A44B
BBLP2	A45C	BBLP3	A46B	BIN10	A82E	BIN2	00B4
BIN20	A859	BINARY	00B0	BINC10	9D11	BINCHK	9CFC
BINFP	A822	BINFP2	A827	BINLP1	9D02	BINLP2	9D13
BINLP3	9D24	BINMIN	00AC	BINOK	9D0A	BITBIN	00A4
BITBN2	00A8	BITERR	A395	BITINT	009D	BITMSG	BC91
BLKBUF	0528	BLNK15	A26E	BLNKS	A270	BOERR	9D20
BOMSG	BC5C	BOOT?	0009	BRKKEY	0011	BRKLST	AA45
BSTEP	000A	BTSMMSG	BD20	C1PTB	BA3C	C65536	BA30
CALPTR	00C9	CALSTK	0480	CDTME3	022A	CELMG	BD5A
CFLD02	9F4D	CFLDOR	9FA0	CFLD1R	9FB9	CHRTAB	BAF4
CHS30	A771	CHSLP	9C29	CHSTAT	A75D	CHTAB2	BAFD
CIOCAL	ACA2	CIOIN2	ACFC	CIOIN3	AD0E	CIOINT	ACF6
CIOTAB	BB07	CIOV	E456	CIX	00F2	CLEMSG	BCC0
CLFMSG	BCCA	CLNLP	9C53	CLNUM	9C4F	CLOSE	000C
CLPRDG	0012	CLRPTR	008C	CLRTAB	009E	CLS	007D
CLSLP1	A5A0	CLST10	A59C	CLSTLP	A5AF	CN2MSG	BD44
CN3MSG	BD4F	COLCMD	0016	COLCRS	0055	CONERR	ADAB
CONFLG	00CA	CONV10	AD94	CONVRT	AD79	CR	009B
CRSINH	02F0	CRYCHK	A986	CRYMSG	BCA9	CRYSND	A98B
CTLR	A26A	CTLR16	A268	CTLRS	053C	CURPRI	009A
DALG	0002	DBITS	0013	DDEC	000B	DDEG	0007
DELCHR	00FE	DELETE	001F	DELLIN	009C	DENTER	0021
DFIX	0019	DFVDUE	001B	DG10	9C65	DG20	9C92
DQ30	9CA4	DQ40	9C8B	DG60	9CBE	DG70	9CD4
DG80	9CDF	DGLP1	9C85	DGLP2	9C90	DGLP3	9C9E
DGRTN	9CE1	DHO10	A0C1	DHOCHK	A0B1	DHOERR	A0BF
DHOFLG	00B7	DHOOK	A0D1	DIGMSG	BC9D	DIGRT	00F1
DIVI	ADEA	DM10	ABA6	DMELP	A5D5	DMEMAL	A5CC
DNAROW	001D	DOLOP	A8D2	DSCAL	AB03	DSCAL2	AAFE
DSOME	A1E8	DSPALL	AAA6	DSPCLR	A578	DSPFLG	00BD
DSPLOD	A559	DSPM3	A89D	DSPMEM	A872	DSPRG	9C5A
DSPROG	9931	DSPRT2	A1E7	DSPRTN	9D7A	DSPST	A549
DSPST2	A54B	DSPSTK	A792	DST10	A540	DSTLP	A535
DUEFLG	00C5	EEXP	00ED	ENDL15	9ADB	ENDLP3	9A25
ENDLP4	9A30	ENDSKP	9A39	ENDWLP	9A21	ENTER	0021
ENTFLG	00C6	EDF	0003	EPERR	9C60	EPMSG	BCB6
EQUAL	008C	ERRCAL	AB95	ERRFLG	00C7	ERRMSG	BDB0
ERRRTN	9BD5	ERRSB2	9BD7	ERRSUB	9BB7	ERRTBL	BC42
ESC	001B	ESIGN	00EF	EXEC	0002	EXEC10	9A70
EXEC20	9A50	EXP	DDC0	EXP1	DE03	EXP10	DDCC
EXP11	DE12	FABCD	9F0B	FACTOR	0025	FADD	DA66
FAHMSG	BD5E	FASC	D8E6	FCHRFL	00F0	FCLOSE	ACF0
FDIV	DB28	FDLP2	9DBA	FDLP3	9DA5	FDLP4	9EF9
FDS05	9DDA	FDS1	9DCA	FDSCOM	A1EE	FDSP0	9D57
FDSP1	9D7B	FDSP2	9D80	FHALF	DF6C	FIXMSG	BD16
FIXNUM	00C0	FLDOM	9F12	FLDOP	DD8D	FLDOR	DD89
FLDOS	9F18	FLDOT	9F49	FLD1P	DD9C	FLD1R	DD98
FLD1T	9F4F	FLPTR	00FC	FMOVE	DD86	FMUL	DADB
FMVPOP	9F9A	FOP20	AC74	FOP30	AC8D	FOPEN	AC60
FOPLP1	AC66	FP10	9CF0	FP20	9CFC	FP21	9D56
FP9S	DFAE	FPBIN	9CE6	FPBNCK	9D2E	FPC10	A002
FPI	D9D2	FPOP	9FA3	FPOPO	9F9D	FPOP1	9FB6
FPOP10	9FB7	FPOPLD	9FA9	FPPTT	009B	FPREC	0006
FPSCR	05E6	FPSCR1	05EC	FPSH05	9FC1	FPSHLD	9FF3
FPSLEN	002A	FPSTK	0600	FPTR2	00FE	FPUSHO	9FBB

COLLEEN CALCULATOR, BY C SHAW

FPUSH1	9FEB	FPX	055C	FRO	00D4	FR1	00E0
FR2	00E6	FRE	00DA	FRX	00EC	FSCR	05E6
FSCR1	05EC	FSLOP	9F66	FSPMSG	BD3B	FSQR	B1D0
FSTOP	DDAB	FSTOR	DDA7	FSTOT	9F55	FST1R	9F60
FSI1T	9F5C	FSUB	DA60	FTEMP	0556	GETC05	A031
GETC06	A06B	GETC07	A077	GETC10	A081	GETC12	A0B9
GETC15	A0BD	GETC20	A094	GETC30	A099	GETCHR	0007
GETDHD	A0A5	GETINT	A0D4	GETMN	A3B4	GETPRI	A106
G105	A0EF	G120	A0FC	GINT2	A0F2	GNOCR	A05D
GOCRY	A6BB	GPR10	A111	GPR20	A123	GTCHR	A026
ICAX1	034A	ICAX2	034B	ICBAH	0345	ICBAL	0344
ICBLH	0349	ICBLL	0348	ICCOM	0342	ICDNO	0341
ICHID	0340	ICPTH	0347	ICPTL	0346	ICSPR	034C
ICSTA	0343	IFP	D9AA	INBUFF	00F3	INIT	9BB4
INIT2	984D	INIT3	AA14	INIT4	989D	INPUT	0004
INS10	AC4D	INS30	AC59	INSEHR	00FF	INSERT	0034
INSLIN	009D	INT2	A6A4	INTADD	A953	INTCHR	BAA2
INTFLG	00A1	INTLBF	DA51	INTMOD	A6E0	INTMUL	AB84
INTRTN	A6AF	INTSUB	A975	INV10	9C35	INVID	9C27
INY	9EBE	IDCB	0340	IOCBSZ	0010	IOERR	ACA9
IOERR2	ACAA	JMPLOP	992E	JMPTBL	BB13	JMPTR1	0090
JMPTR2	0092	KBUFF	BAEF	KEY20	9AD5	KEY40	A310
KEY50	A312	KEY60	A31C	KEYCHR	0088	KEYCNT	00BF
KEYERR	9BB5	KEYLEN	0089	KEYLN2	00BA	KEYLP1	9AA0
KEYLP2	9AA5	KEYMSG	BC4C	KEYWRD	BE0B	KIOCB	0010
KYLFRT	008E	LBPR1	057E	LBPR2	057F	LBUFF	0580
LD1DIV	ADB6	LD1MUL	ADC5	LDCH05	A145	LDCH10	A147
LDCHR	A124	LDCAV	00C4	LDV	A39A	LDI	A39E
LDINT	A1B4	LDN	A3A8	LDN20	A15C	LDNBSV	00BB
LDNIB	A14C	LDPMT	A3AC	LDPV	A3B0	LDY1DV	ADB4
LDY1ML	ADC3	LENERR	9B6A	LENG	AD47	LENGTH	BA42
LEX	9A51	LEX30	9A92	LEXERR	9B6C	LEXRTN	9B7F
LFAROW	001E	LFRT	0080	LINLEN	0026	LIST	003A
LMARG	0001	LMARGN	0052	LOAD	003C	LOG	DECD
LOG10	DED1	LOG10E	DEB9	LOGCHK	A6C7	LOGCLP	A6D0
LOOP	98C7	LOOP3	98E5	LOOP4	98F9	LOP10	ABF5
LQP20	ABFE	LOP30	A900	LOPLP1	ABD9	LOPLP2	ABEA
LPAD	00BD	LPAR	008A	LX50	9B34	LX60	9B63
LXERR2	9AF4	LXGT2	9B3B	LXHVDT	9B12	LXINIT	9A3C
LXLP20	9AB5	LXLP40	9AFD	LXN2	9B48	LXN3	9B5D
LXND2	9B1D	LXNDOT	9AFB	LXNMCK	9AE5	LXNUM	9B45
LXRTN2	9BB5	LXRTN3	9BB0	MAIN02	9BD2	MAIN04	996A
MAIN05	996D	MAIN10	A01A	MAIN15	9B9D	MAIN20	9BA3
MAIN21	9BB0	MAIN35	998D	MAIN40	9990	MAIN50	999E
MAIN60	99A9	MAIN62	99B7	MAIN65	99BD	MANTLN	00C1
MAS	AD59	MASS	BA6C	MATCH	AB40	MEMA10	AFB7
MEMA20	AFBD	MEMA30	AFCD	MEMADD	AFAA	MEMADR	00CE
MEMCLR	AA03	MEMDIV	AFCE	MEMFLG	00CB	MEMLDO	A3B2
MEMLD1	A3B8	MEMLD2	A3BE	MEMLDR	A3C0	MEMLEN	0064
MEMMSG	BD2C	MEMMUL	A3E6	MEMNUM	00A3	MEMSTQ	AE4A
MEMSUB	A3D0	MINUS	0089	MLD10	A3CA	MLD20	A3CE
MOD	0041	MODFAC	0550	MS10	A3DB	NATCF	000B
NCHKLD	A161	NCK10	A176	NCK30	A1B3	NCK40	A1B4
NEGFLG	00A0	NERR	A9C4	NOBOOT	9B5A	NDEXEC	9A75
NOMAT	AB1C	NOPFLG	0097	NORM	DC00	NOSNER	B0B5
NOST10	994E	NOSTOR	993D	NSCF	0006	NSEMSG	BC6B
NSFMSG	BC74	NSIGN	00EE	NUMBER	00BE	NUMFLG	00A2
NUMLEN	000E	OFFERR	A744	ONE	BA42	ONEADD	A951
ONESUB	A973	OPEN	0003	OPFLG	0096	OPPTR	009C
OPSADR	00CC	OPSLN	0100	OSEMSG	BC7F	OSFMSG	BC8B

NUMLEN	000E	ONE	BA42	ONEADD	A951
ONESUB	A973	OPFN	0003	OPPTR	009C
OPSADR	00CC	OPSLN	0100	OSEMSG	BC7F
				OSFMSG	BC8B

COLLEEN CALCULATOR, BY C SHAW

OUTPUT	000B	PAND	0006	PBUFF	BAB1	PC	00B9
PC1MAX	00D2	PC1MX1	00D3	PCADD	A19F	PCADD1	A1AD
PCADDN	A19D	PCINC	A199	PCLRO	A1B0	PCN05	A191
PCN10	A194	PCN20	A19B	PCNCHK	A185	PEQUAL	0001
PHIGH	000D	PICONS	BA00	PIDCB	0020	PIDV18	BA36
PIQV4	DFF0	PIOV1	A3F1	PKPTR	008C	PLPAD	0000
PLRPAR	0002	PLUS	008B	PLYARG	05E0	PLYEV1	DD40
POP10	A1D1	PDP20	A1D4	POPC10	ABDF	POPCAL	ABD6
POPNI0	A74D	POPNI20	A751	POPDP	A1C4	POPRTN	A752
POR	0005	POWER	0051	PPLUS	0007	PPOWER	0009
PREVOP	009B	PRGADR	00D0	PRGLEN	0400	PRIOTB	BF64
PRNCHK	A2CC	PRNFLG	0095	PROG	00BB	PRDGRA	0053
PROMSG	BD00	PRVPR1	0099	PRVSTK	0565	PSET0	A1B9
PSH10	A1E5	PSHC10	AB9B	PSHC20	AB9C	PSHCAL	ABBE
PSPEC	000E	PSPEC2	000A	PTABC	A2B3	PTABD	A2B6
PTCHR	A231	PTCHR2	A233	PTCHS2	A27C	PTCHSP	A2AB
PTCRPD	A20B	PTCRPN	A297	PTDEL2	A22B	PTIMES	000B
PTLIN1	A249	PTLP	AAC0	PTMSG2	9BF2	PTTXXP	A205
PUSHOP	A1D7	PUTBLK	9F6F	PUTBRT	9F6E	PUTCHR	000B
PUTCHS	A274	PUTCMD	9CC0	PUTCR	A20B	PUTCR2	A21D
PUTCRP	A29D	PUTCTL	A272	PUTDEL	A227	PUTMSG	9BF0
QUADFL	00B8	RADFLG	00FB	RADPI2	BA24	RAMCLR	AA07
RAMSET	AA09	RANDOM	D20A	RETNI	A2B5	RETURN	A49B
RF10	9E07	RF100	9E58	RF105	9E6C	RF110	9E75
RF120	9E91	RF130	9EA2	RF140	9EAB	RF142	9EC5
RF14B	9ECE	RF150	9EDA	RF160	9EE9	RF170	9EF1
RF20	9E0F	RF30	9E11	RF40	9E1B	RF50	9E20
RF70	9E02	RF80	9E29	RF85	9E47	RF90	9E55
RFERR	9E63	RFLP1	9E31	RFLP2	9E5A	RFLP3	9E83
RFLP4	9E93	RFLP5	9EE2	RMARG	0026	RMARGN	0053
ROWCMD	0016	ROWCRS	0054	ROWREG	0005	ROWSCR	0010
ROWSTT	0001	RPAR	00BB	RPNALG	0094	RST	0063
RTAROW	001F	S1B0PI	A4B4	S2CMP	A406	SABSV4	B249
SAC10	A4A0	SAC30	A4A6	SAC34	A4CB	SACDS	A49C
SADV	A71B	SADV10	A71C	SADV20	A726	SALG	A779
SALGN	A77D	SAND	ABC8	SAS10	B11B	SASIN	B105
SATAN	B11B	SAVCHR	A2D7	SAVE	0065	SAVL0D	AD14
SBAL	AE1B	SBAL05	AE4B	SBAL15	AE20	SBAL20	AE23
SBCL5	A00C	SBERR	A4F1	SBITS	A4E0	SBITS2	A4F4
SBLP1	A509	SBLP2	A515	SBLP3	A527	SBST05	A9A1
SBST10	A9A3	SBST30	A9BB	SBST40	A9C7	SBST50	A9CB
SBSTEP	A990	SC	A940	SCAL10	ABB9	SCAL20	ABC3
SCAL30	ABD3	SCALL	AB9F	SCDEG	ADBC	SCH10	A5BA
SCHGSG	A5B2	SCLCAL	AB84	SCLINI	A5BB	SCLMEM	A5C9
SCLP2	A3FC	SCLPRD	A9F5	SCLR	A774	SCLSTA	A59B
SCLSTK	A7BA	SCLX	A1B0	SCM	AD41	SCMP2	A3FA
SCMPND	ADF3	SCDEF	BA06	SCOMPL	A57F	SCONFG	00CB
SCONTI	BCF0	SCORRE	AFF5	SCDS	B276	SCRAD	ADAD
SCUP	AD69	SDEC	A704	SDEG	A753	SDEL2	AC10
SDELET	ABFE	SDELP1	AC1B	SDELP2	AC09	SEFORM	00BF
SEND	AA93	SENDER	AE0B	SETMSG	9C06	SETRTN	9C26
SETTAB	009F	SETVBV	E45C	SEXP05	9833	SEXP10	9842
SEXPE	9800	SEXPRT	9841	SEXPTE	9807	SF	ADCB
SF10	A5FA	SFACT0	A5E9	SFADD	A96A	SFDIV	A93A
SFDON	A615	SFERR	A612	SFERR2	9847	SFIND	AE0F
SFIX	A55D	SFIX2	A56B	SFLOZ	AD5D	SFLP	A5FE
SFMUL	AB97	SFND10	AE11	SFND20	AE15	SFRACT	A97A
SFVUB	A9B3	SFT	AD35	SFV	B250	SFV05	B271
SFV20	B263	SFVDUE	ADF7	SFVORD	ADFB	SGAL	AD71
SGM	AD57	SGD1	AB54	SGD2	AB4F	SGDERR	AB7B

COLLEEN CALCULATOR BY J SHAW

BOOTO	AB4D	BHXL	A70B	SHFOS	A7FD	SHF10	AB04
SHF13	AB07	SHF25	AB15	SHF30	AB29	SHF40	A7ED
ST	AB24	STOS	AB43	ST15	ACT9	STC019	STB1
STN	AB31	STADON	STF2	STINER	ST02	STIN1	ST06
STNFC	STC7	STNFA	STET	STN1N	ST04	STANND	ST1B
STNFR	AC34	STNLP	AC3C	STNTEG	AB83	STOCB	ST00
STAD	AD4B	STN	AD43	ST	AD75	STASH	ST07
STL	AD53	STD10	AFDA	ST1ST	AA22	STLN	AB83
STLAD	AC77	STOSTE	ABPE	STLSC5	A41C	STSHF	A7DA
STSHF2	A41B	STSTO3	AA3E	STSTO7	AA41	STST10	AA4B
STSTLP	AA25	SM	AD2D	SM1	AD3D	SHOD	A6EB
SHSD	STC3	SM	AE9B	SNOS	AEES	SH20	AEBE
SH30	AED9	SH50	AE5A	SHDLP1	9C45	SHOP	AA92
SHOTRA	BCDC	SHUM20	A436	SHUM25	A477	SHUM30	AAB7
SHUM40	A4BB	SHUM50	A42A	SHUMB	A42B	SHWE10	B063
SHCF	A70C	SHCT10	A70E	SHOFF	A72E	SHON	A737
SHR	ABCC	SHRD10	ADFD	SHOUND	9C3A	SHOZ	AD4F
SHADD	A967	SHPAULP	AA5E	SHPAUSE	AA4B	SHPCEND	BAEF
SHPLEN	000A	SHPCTBL	BAE4	SHPDIV	A937	SHPI	A4BD
SHPI10	A4C4	SHPT	AF1E	SHPT05	AF61	SHPT20	AF29
SHPT20	AF44	SHMUL	AB94	SHPOLAR	B17E	SHPO	9F9D
SHPOPC	AB8B	SHPOW25	A626	SHPOW30	A629	SHPOW40	A62C
SHPOW50	A650	SHPOW60	A615	SHPOW70	A674	SHPOW80	A677
SHPOWER	A619	SHPRINT	A90B	SHPROGR	AA66	SHPROLP	AA7A
SHSUB	A980	SHUSH	9FBB	SHV	AF66	SHV05	AF92
SHV20	AF79	SHVDUE	AE03	SHVORD	AE07	SHGR0	B1B5
SHR1	B1C8	SHGR2	B23B	SHGR3	B241	SHGRDON	B21E
SHRLP	B1E7	SHROUT	B249	SHGT	AD6D	SHRAD	A757
SHRAD10	A759	SHRCL	9F7F	SHRECIP	A924	SHRECTA	9F1D
SHRESET	AABA	SHRET10	ABFB	SHRET20	ABFD	SHRETUR	ABE6
SHROOT	A616	SHROU10	A671	SHROUND	A660	SHRPN	A7B1
SHRPN10	A7B3	SHRSHE	A7DE	SHRTN	A429	SHRUN	BCED
SHSAV10	ACEB	SHSAVE	ACDF	SHSIGN	00C2	SHSIN	B06A
SHSIN2	B06D	SHSLOPE	B03A	SHSMINU	BFA8	SHSPLUS	BFAF
SHSQRT	B1A7	SHSSUAR	AB5A	SHSSTEP	A9CC	SHSTEP	006C
SHSTFLQ	00BE	SHSTO	AB60	SHSTO10	AB65	SHSTOLD	0566
SHSTP	AA93	SHSTP10	A9D9	SHSTP15	A9EF	SHSTP20	A9F2
SHSUM	ABA7	SHSTACHR	BAB7	SHSTAN	A92B	SHSTAR	0086
SHSTARS	BAFO	SHSTART	9B4A	SHSTATLN	BDB6	SHSTBSP	AD65
SHSTCRTN	A417	SHSTKD10	A7A2	SHSTKD30	A7C1	SHSTKD40	A7D3
SHSTKD45	A7D4	SHSTKLIN	BDF1	SHSTLN2	BDD4	SHSTMSG2	9C04
SHSTOPRG	0001	SHSTP	006E	SHSTPR40	9924	SHSTR10	BCE2
SHSTR15	BCEB	SHSTR20	BCEC	SHSTRACE	BCE0	SHSTRUNC	A67D
SHSTSP	AD61	SHSUBCAL	A005	SHSUBINT	A68F	SHSUBONE	A68D
SHSUCCES	0001	SHSX	B00A	SHSXCHGY	9FEB	SHSXCHM	ABB3
SHSXEQ	AB15	SHSXGE	AB23	SHSXL	AB2E	SHSXMEAN	AFD9
SHSXNE	AB39	SHSXOR	ABD0	SHSXRI0	A917	SHSXRTN	ABA6
SHSXSTDD	BFEC	SHSXVARI	AFE1	SHSY	A95B	SHSYD	AD39
SHSYINTE	B01C	SHSYMEAN	AFDD	SHSYSTDD	BFF2	SHSYVARI	AFE5
SHTO	009E	SHTL	009F	SHTAB	007F	SHTABLE	BC27
SHTIOCB	0030	SHTKINT2	9C22	SHTKBUF	0500	SHTKCHR	BACC
SHTKCLN	000D	SHTKCOD	00B1	SHTKEND	BADA	SHTKINT	9C20
SHTKLEN	002B	SHTKN10	9DA3	SHTKNUM	9DB6	SHTKPTR	00B2
SHTKTBL	BADA	SHTKTIN	00B6	SHTKTMP	00B4	SHTKMSG	BC42
SHTRACE	00BC	SHUKERR	A37E	SHUNIMSG	BCD4	SHUNKRTN	A36A
SHUNKY10	A35C	SHUNKY20	A373	SHUNP10	A32E	SHUNPACK	A323
SHUNPCK2	A31F	SHUNPINT	A2E2	SHUNPKEY	A347	SHUNPLP	A35F
SHUNPNUM	A2F3	SHUNPNXT	A2FF	SHUPARDW	001C	SHVOL	AD77
SHVOLUME	BA7E	SHWARM	9B97	SHWARMST	000B	SHWLOOP	99CC
SHWLP05	99EC	SHWLP10	99F5	SHWLP20	99FF	SHWLP30	9A10

COLLEEN CALCULATOR BY C SHAW

XCLOSE ACF2 XEFORM D920 XEFORM DE95 YINT A495

UNPNUM	A2F3	UNPNXT	A2FF	UPAROW	001C	VOL	AD77
VOLUME	BA7E	WARM	9897	WARMST	0008	WLOOP	99CC
WLP05	99EC	WLP10	99F5	WLP20	99FF	WLP30	9A10

COLLEEN CALCULATOR, BY C SHAW

XCLOSE	ACF2	XEFORM	D920	XFORM	DE95	XINT	A695
XINT1	A6A0	XINT4	A680	XLERR	AB14	XLTSUB	AB06
XSAVE	0563	YSAVE	0564	Z11IMN	A970	Z11INI	ADE7
Z1ILDN	A3A2	Z1IMN	A921	Z1IN	ADDB	Z1INM1	ADE1
Z1PLI	A94E	ZDEG	BD99	ZEND	0020	ZFL	BDAA
ZKG	BDA5	ZLN1I	A6B0	ZM	BDA2	ZMRTN	ABA6
ZMUL1I	A8B9	ZPOLAR	BD7D	ZRAD	BD9D	ZRECT	BD61
ZRDPG	00B0	ZSIGMA	AF97	ZTEMP1	00F5	ZTEMP3	00F9
ZTEMP4	00F7	ZVAR	AFE7	ZVAR2	9FC9		